ED 014 071 AL 000 795

DEUXIEME CONFERENCE INTERNATIONALE SUR LE TRAITEMENT AUTOMATIQUE DES LANGUES (SECOND INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, GRENOBLE, FRANCE, AUGUST 23-25, 1967).

PUB DATE AUG 67

EDRS PRICE MF-$2.00 HC-$20.00 498P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *MACHINE TRANSLATION, ALGORITHMS, GRAMMAR, *CONTRASTIVE LINGUISTICS, CHINESE, ENGLISH, TRANSFORMATION THEORY (LANGUAGE), RUSSIAN, PHONOLOGY, MORPHOLOGY (LANGUAGES), FRENCH, GERMAN, PHRASE STRUCTURE,

THE PAPERS IN THIS VOLUME--CATEGORIZED AS EITHER (1) AUTOMATIC ANALYSES OF NATURAL LANGUAGES, (2) STATISTIC AND SEMANTIC ANALYSES OF LINGUISTIC DATA, OR (3) ALGEBRAIC THEORIES OF LANGUAGES--ARE THE COLLECTIVE PRODUCT OF THE INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS HELD AT GRENOBLE, FRANCE, AUGUST 23-25, 1967. TOPICS RANGE FROM ALGORITHMS FOR DEPENDENCY GRAMMARS AND SLANT GRAMMAR CALCULUS TO AUTOMATED FIELD METHODS, AUTOMATIC TRANSLATION, AND CRITICAL COMMENTARY ON PREVIOUS LINGUISTIC WORK. THE THIRTY-NINE PAPERS REFLECT CURRENT THOUGHT AND TRENDS IN COMPUTATIONAL LINGUISTICS IN FRANCE, GERMANY, RUMANIA, THE SOVIET UNION, AND THE UNITED STATES. ALL ENTRIES ARE IN EITHER ENGLISH OR FRENCH. INFORMATION CONCERNING THE AVAILABILITY OF THIS DOCUMENT MAY BE OBTAINED FROM C.E.T.A., BOITE POSTAL NO. 8, 38 SAINT-MARTIN-D'HERES, FRANCE. (FB)

ED014071

2ème CONFERENCE INTERNATIONALE SUR LE TRAITEMENT AUTOMATIQUE DES LANGUES

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

AL 000 795

GRENOBLE 23 - 25 août 1967

"PERMISSION TO REPRODUCE THIS ~~COPYRIGHTED~~ MATERIAL HAS BEEN GRANTED
BY A. Hood Roberts

TO ERIC AND ORGANIZATIONS OPERATING UNDER AGREEMENTS WITH THE U.S. OFFICE OF EDUCATION. FURTHER REPRODUCTION OUTSIDE THE ERIC SYSTEM REQUIRES PERMISSION OF THE ~~COPYRIGHT~~ OWNER."

TABLES DES MATIERES

-I- ANALYSE AUTOMATIQUE DES LANGUES NATURELLES

1	COYAUD M.	Problèmes d'analyse automatique du chinois
2	DANIEL (Mc) J. PRICE W.L. SZANSER A.J. YATES D.M.	An evaluation of the usefulness of machine translations produced at Teddington, and an account of the translation methods.
3	ENGELIEN G.	Programming of reversible systems in computational linguistics.
4	FRASER KLIMA PETRICK	A formalisation of the transformational subcomponent of transformational grammars.
5	GARVIN P.	The Fulcrum syntactic analyser for Russian.
6	GUIERRE L.	Analyse automatique du mot anglais en phonologie et morphologie.
7	HARPER K.	Syntactic and semantic problems in automatic sentence generation.
8	JOSHI A.K. HIZ D.	Transformational decomposition.
9	KARLGREN H.	Slant grammar calculus.
10	KAY M.	Experiments with a powerful parser.
11	LAZSLO S.I.	The relevance of some compiler construction techniques to the description and translation of languages
12	MASTERMANN M.	Man-aided computer translation from english into french using an on-line system to manipulate a bilingual conceptual dictionary or thesaurus.
13	MEYER-INGWERSEN J.	Analyse de propositions allemandes.
14	REWZIN	Some linguistic arguments in favour of Mr Bar-Hillel's "non uniqueness" hypothesis".
15	RICHTER H. HINCHA G.	Can overt syntactic structure be interpreted as a logical device ?
16	ROBINSON J.	Two algorithms for obtaining corresponding phrase structure and dependency grammars.
17	SALKOF M. SAGER N.	The elimination of grammatical restrictions in a string grammar of english
18	SAMUELDORFF P.	The application of Fortran to automatic translation.
19	SHILLAN D.	Segmenting natural language by articulatory features.

20	VAUQUOIS B. VEILLON G.	Un métalangage de grammaires transformationnelles.
21	YATES D.M.	A computer model for Russian grammatical description and a method of English synthesis in machine translation.
22	ZALIZNIAK	Un modèle de la notion de cas.

-II- ANALYSE STATISTIQUE ET SEMANTIQUE DES DONNEES LINGUISTIQUES

23	BACHRACH J.A. HIRSCHBERG L.	Une troisième version du Dicautom.
24	BENZECRI J.P. ESCOFFIER B. FENELON P.J. ROUX M.	Classification automatique et analyse des tableaux de données
25	BORKOWSKI C.	An experimental system for automatic recognition of personal names in newspaper texts.
26	BORILLO A. VIRBEL J.	Utilisation d'un système de relations syntaxiques pour l'indexation automatique des documents
27	EDMUNDSON H.P.	Axiomatic characterization of synonymy and antonymy.
28	GUNTHER K. SCHWEISTHAL	Semantically governed machine translation of BE-clauses with adverbs and prepositional phrases.
29	HOPPE A.	Mechanical explication and translation of the semantic factor "ownership" in a german Zuwendsatz".
30	JOLKOVSKI A. MELCHOUK I.	Essai d'une théorie sémantique applicable au traitement de langage.
31	KLEIN G.	Autoling : an automated linguistic fieldworker.
32	MARTEMIANOV Y. MOUCHANOV Y.	Une notation sémantique et ses implications linguistiques.
33	OLNEY J. REVARD C. ZIFF. P.	Some computational aids obtaining a formal semantic description of english.
34	PAGE E.	Statistical and linguistics strategies in the computer grading of essays.
35	PENDERGRAFT E.D.	An experiment in automatic classification of english.

-III- THEORIE ALGEBRIQUE DES LANGAGES

36	ABRAHAM S.	An integrated language theory.
37	BENZECRI J.P. FRIANT J. GUEDJ R. ROUBAUD J.	Nouvelles recherches sur la théorie algébrique des langues.
38	BRODDA B.	The entropy of recursive Markov processes.
39	MARCUS S.	Categories de Dobrusin, fermetures de Sestier et voisinages de Sakaï.

Com/P 4/1967/MC-Grenoble

Section d'Automatique Documentaire
du
CENTRE NATIONAL DE LA RECHERCHE
SCIENTIFIQUE

Analyse et classification automatiques en chinois

par Maurice Coyaud

(Communication au Congrès International de Linguistique mathématique - Grenoble, Août 1967 -)

On se propose d'abord une rapide revue des travaux soviétiques et américains concernant l'analyse automatique des textes scientifiques chinois ; une seconde partie de la communication est consacrée à un projet de classification nominale (par les particules "numérales" : measure words, classifiers) à l'aide de machine.

1. Essai de synthèse des recherches en analyse automatique.

Dans un premier temps, les recherches ont concerné l'analyse dépendante (ou binaire) des textes chinois : cette analyse était effectuée en fonction de la langue de sortie : l'anglais (travaux de Reifler, Washington), le russe (travaux de Zvonov, 1961).

Mais on s'est aperçu qu'il était plus rentable et plus intéressant de considérer le problème de l'analyse automatique sans tenir compte de la langue de sortie. C'est principalement ces recherches, sur l'analyse indépendante, qui retiendront l'attention (néanmoins, on tiendra compte des remarques pertinentes de Zvonov, issues de son travail sur l'analyse dépendante du chinois, en particulier sur les constructions par encadrement : ramočnye konstrukcii).

Les problèmes de l'analyse syntaxique du chinois ont été définis dans leurs grandes lignes par Voronin (1958), Wang (1963) et Hashimoto (1965). Selon Voronin, on commence par découper la phrase, si elle est complexe, en propositions simples ; on établit ensuite les groupes syntagmatiques et enfin les fonctions logiques : sujet et prédicat. La phrase complexe est décomposée au moyen de bornes formelles comme la ponctuation, certaines conjonctions ; certaines propositions n'ont pas de frontières formellement bien déterminées : par exemple, les prépositives (ou déterminatives développées avec *de* : cf. Nikitina, 1965, et le rapport sur la technique Fulcrum de Garvin, 1965), et les relatives sans mot de relation ; ces dernières sont de deux sortes : ou bien introduites par des verbes comme penser, (re)connaître, etc. ; ou bien, en elles, le mot relié au verbe précédent en qualité de complément, sujet inversé ou partie nominale du prédicat est en même temps le sujet d'un groupe de mots plus éloigné. Avant de procéder à la synthèse de la traduction en langue de sortie, l'analyse devra spécifier le nombre des noms (catégorie non obligatoire, et peu fréquente en chinois) et le temps des verbes (compte tenu des cinq aspects du verbe chinois, des verbes modaux et des expressions de temps).

La catégorisation en parties du discours joue le rôle essentiel dans ces opérations et on la justifie en détail (Žerebin 1961 a, Voronin 1961) ; la nature du corpus joue aussi son rôle, et notoirement les difficultés introduites par la présence de *wényán* dans les textes de mathématiques, apportant une concision qui complique la tâche de l'analyste, puisque le système grammatical *wényán* ne s'incorpore pas, mais se superpose au système *bǎihùa*.

A côté des problèmes généraux qui font l'objet des articles qui viennent d'être cités, on trouve un nombre appréciable de monographies étudiant notamment :

1) les ambiguités de catégorie grammaticale (Žerebin 1961 b : réduction de 5 à 3 types d'ambiguité, en raison du passage d'un système d'analyse dépendante à une analyse autonome ; Nikitina 1966 ; Zvonov 1961 b : ambiguités des mots-outils *yòng*, *yǐ*, *jiāng*, *yóu*, *wèi*). La résolution de ces ambiguités est jugée trop difficile par Ertel *et al*. qui la supposent résolue avant le démarrage de leur technique d'analyse syntaxique ;

2) les constructions avec *de* (Fulcrum Report, 1965) ;

3) la reconnaissance automatique des groupes nominaux (Peng, 1966) ;

4) la recherche du sujet (Nikitina, 1965 :
place du sujet ; types de sujets : propositionnel, nominal etc... ; doublage du sujet par un pronom ou un nom généralisant ; sujets homogènes ; sujets absents ; sujets de subordonnées de détermination avec *de*) ;

5) l'imbrication (Hashimoto, 1966) ;

6) l'interrogation (Elliot, 1965) ;

7) des déterminants définis et indéfinis (Annear, 1965).;

8) des conjonctions de subordination employées dans les phrases hypothétiques, notamment *rúguǒ*, *tǎngshǐ*, *tǎngruò*, *jiǎshǐ*, *yàoshì* et leurs corrélateurs *nà*, *jiù*, *cái*, *biàn*, *yě* (Coyaud, 1967). Les différents facteurs déterminant

l'emploi de telle ou telle conjonction sont étudiés. Il n'est pas certain que l'emploi de ces conjonctions soit dicté uniquement par des facteurs dialectaux (par exemple *jiǎshǐ* viendrait du Sìchuān, *yàoshì* du Shānsī, les composés de *tǎng* du Zhèjiāng) ; on sait d'autre part que *jiù* s'emploie quand la principale est brève, *nà(mo)* quand la principale est longue ; mais les autres critères, déterminant le choix des corrélateurs n'ont pas été dégagés.

2. Classification nominale.

Le chinois contemporain possède une centaine de particules "numérales" déterminant chacune une classe nominale. Le mot "numérale" est gênant, car ces particules ne s'emploient pas nécessairement entre un nombre et le nom, mais aussi entre un démonstratif et le nom ou des adjectifs précédant ce nom. Ceci dit, nous conserverons le mot "numérale", qui est traditionnel ("classificateur" serait trop général). Le nombre des numérales a été souvent exagéré (plusieurs centaines, pour Zhao Yuanren ou Hockett) mais les inventaires ne visaient sans doute pas strictement l'état de langue du chinois contemporain. Pour le *guóyǔ*, Herrfurth (1964) considère que 48 numérales sont d'usage courant ; un inventaire fondé sur un corpus plus large (Coyaud, 1967 b) relève 84 numérales au sens restreint et 16 numérales indiquant des capacités d'usage ordinaire. Le travail que nous présentons ici a un intérêt propre, qui est théorique ; mais il a aussi un intérêt pratique (§ 2.1).

2.1. Les particules numérales et l'analyse automatique.

En linguistique appliquée, la classification des noms par les numérales a une double importance. D'abord, en pédagogie : il est indispensable d'enseigner avec précision à celui qui veut apprendre le chinois avec quelle(s) numérale(s) chaque nom se combine (l'ignorance de ces combinaisons est aussi pesante que celle du genre des noms en français ou en allemand). D'autre part, en analyse automatique, il est nécessaire que l'automate dispose d'un dictionnaire complet, attribuant à chaque nom la (ou les) numérale(s) qui lui correspond(ent). De tels dictionnaires n'existent encore que de façon partielle. L'information consistant à indiquer les couples numérales-noms dans le dictionnaire automatique a une double utilité :

a) permettre de résoudre des polysémies lexicales ; par exemple *yī gè yuè* "un mois" et *yī lún yuè* "une lune", ne diffèrent que par la particule numérale, *gè* s'appliquant au mois, et *lún* à l'astre. Le mot *yuè* est homographe et homophone.

b) permettre de résoudre des ambiguités de catégories grammaticales. Les particules numérales sont pour la plupart des homographes pouvant souvent fonctionner comme noms et parfois comme verbes (*dài* "emporter", *juǎn* "enrouler", etc.). Un essai d'établissement automatique des syntagmes nominaux (avec particules numérales; nécessairement) a été fait par Peng (1966).

On pourrait ajouter à ces deux rubriques une troisième, représentant un champ d'études encore inexploré : l'utilisation de la classification nominale en tant que groupes d'affinités sémantiques, permettant de choisir entre plusieurs corrélations acceptables d'un point de vue purement grammatical (1).

2.2. Classification nominale à l'aide des numérales (problème théorique):

L'étude préliminaire au projet esquissé ci-dessous fait l'objet d'un rapport détaillé (Coyaud, 1967 b) dont nous nous contenterons ici de développer les implications dans la perspective d'un traitement par machine. Le problème consiste à établir avec détail et précision la classification des noms en chinois, à l'aide des particules numérales. En particulier il s'agit de dresser la liste des ensembles disjoints, celle des ensembles conjoints, déterminer exactement les éléments appartenant à chacun de ces ensembles, et en particulier établir la liste des éléments appartenant aux zones d'intersection entre ensembles pris deux à deux, et n à n. La tâche est assez considérable pour justifier une aide mécanique, si l'on se propose effectivement de classer environ 3,000 noms. Nous avons entrepris le travail à la main pour environ 500 noms. Nous donnerons la physionomie générale de la classification telle qu'elle apparait à l'issue de ce travail. Nous ne nous sommes pas limités à une perspective synchronique (description du système classificatoire des noms du chinois contemporain : CC) ; le problème de l'évolution de ce système est important. On relève en effet peu de numérales en chinois archaïque (cf. Jaxontov et Huáng Zàijūn) ; par contre, en chinois ancien (CA) de la période 200.600 de notre ère, Liú Shìrú a pu relever 141 numérales propres et 54 numérales exprimant des capacités (sans être pour autant des unités de mesure codifiées, qui ne nous intéressent pas ici). En guóyú ou chinois contemporain (à partir de 1920 environ), notre inventaire comprend 100 numérales (84 propres et 16 de capacité). Les deux listes de numérales, pour le CA et le CC, sont données en annexe. Pour le CA, nous n'avons retenu que les 141 numérales au sens propre du mot, qui seront comparées aux 84 numérales propres du CC pour ce qui est de la classification des noms qu'elles déterminent. Les 141 du CA n'incluent pas toutes les 84 du CC. Ces deux ensembles sont en intersection. Les différences individuelles sont données dans notre rapport détaillé. Indiquons ici simplement que sur les 141 du CA, 80 n'apparaissent pas en CC ; inversement, sur les 84 du CC, 24 n'étaient pas attestées dans les corpus de CA compilés par Liú Shìrú.

---

(1) Pour une utilisation d'un réseau d'affinités sémantiques en analyse automatique de textes français, cf. Coyaud, Siot-Decauville, L'analyse automatique des documents, Mouton, Paris, 1967.

2.21. Relations entre classes nominales en CA.

Les ensembles simples disjoints, au nombre de 11, sont définis par les numérales : *fán*, *qǔ*, *jiàn* 2, *kǒng*, *luǎn*, *niǔ*, *pò*, *qì*, *tà*, *wén*, *zhí* 2. Chacun des 130 autres ensembles est en intersection avec au moins un ensemble de ce groupe. Il y a quatre groupes d'ensembles disjoints parmi ces 130 (l'élément commun est indiqué après) :

a) *bǐng* 1, *fú* 2 : éventail

b) *chàng*, *gēng*, *kè* : unités de temps

c) *jié* 1, *jù* 1 : verset

d) tous les 123 restants. Le diagramme donné en annexe contient ces ensembles. Une seule intersection a été indiquée à chaque fois (afin d'éviter l'encombrement). Bien d'autres intersections sont attestées. Par exemple, nous n'en avons donné qu'une seule pour *gè*, alors qu'en CA, d'après les exemples de Liú Shìrú, l'ensemble défini par *gè* est en intersection avec quinze autres ensembles, définis par les numérales suivantes (l'élément commun est donné après) :

míng, *yuán*, *jiè*, *wèi* 1, *rén*, *tóu* : humains
fā, *jù* 2, *zhī* 3 : flèche
shù, *běn*, *zhū*, *shuāng*, *cóng* : arbres
zuò : statue de divinité

2.22. Relations entre classes nominales en CC.

En CC (*guóyǔ*) les résultats fournis par notre enquête donnent 19 classes disjointes et 80 classes en intersection, dont les éléments forment un graphe connexe (voir le second diagramme). L'ensemble défini par *gè* est en intersection avec 25 ensembles (ces intersections ne sont pas toutes indiquées dans le diagramme), définis par les numérales suivantes (les éléments communs sont donnés dans Coyaud 1967 b) : *bǎ*, *bù*, *chuáng* 2, *dào*, *dǐng*, *gēn*, *jiē*, *jià*, *jiān*, *jiàn* 1, *kē* 1, *kuài*, *liǎng*, *miàn*, *míng*, *piàn*, *suǒ*, *tiáo*, *tóu*, *wèi* 1, *yuán* 1, *zhī* 3, *zhēng*, *zhǒng*, *zuò*.

2.23. Le *gèhuà*.

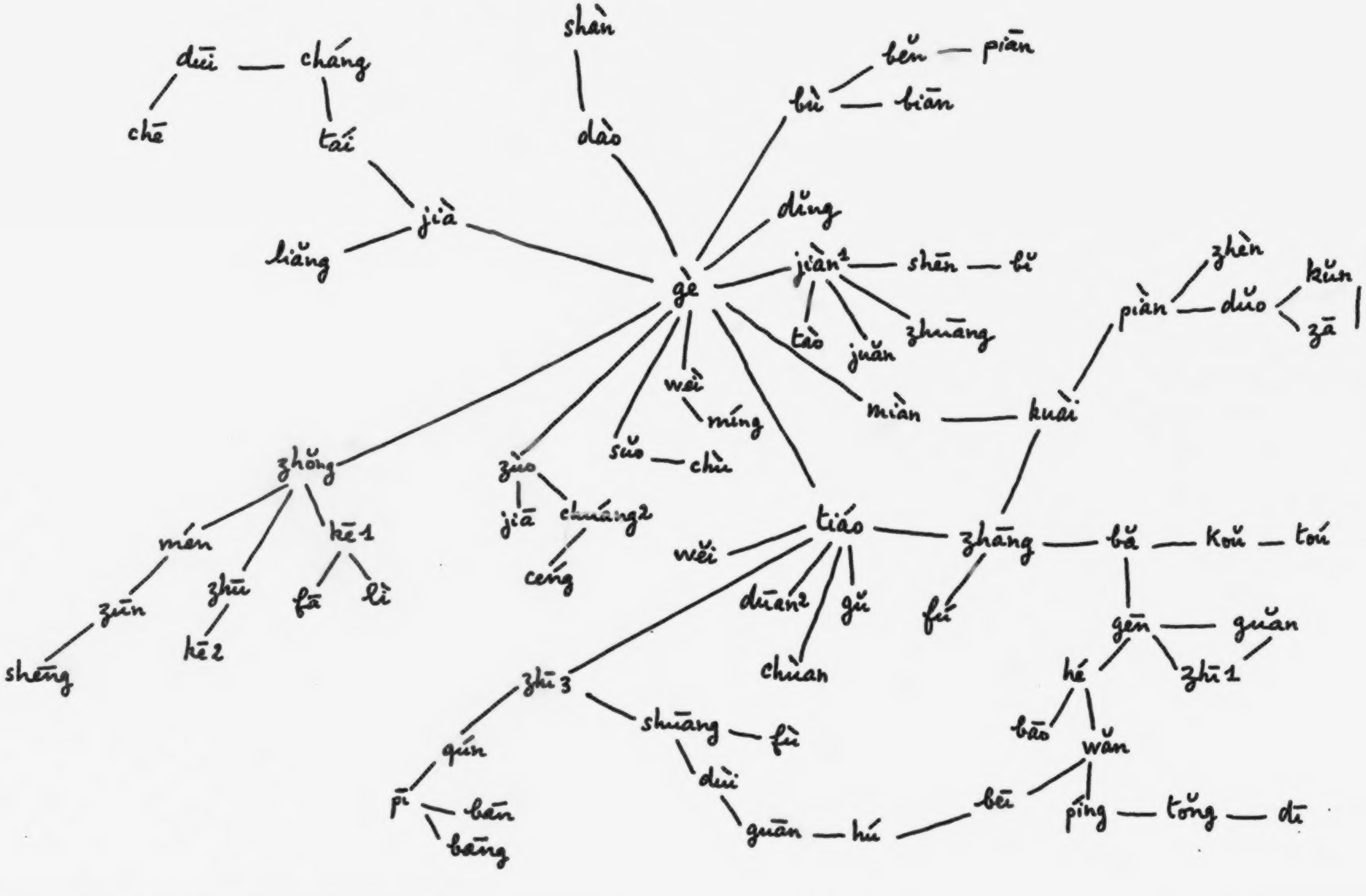
L'importance exacte du *gèhuà* est difficile à évaluer d'après ce sondage sur la classification de 500 noms à peine. Signalons que *méi* en CA semble jouer le rôle de *gè* en CC. L'augmentation du nombre des substitutions par *gè* de numérales du CA au CC n'est pas énorme, elles sont passées de 15 à 25. Mais 6 numérales seulement sont identiques dans les groupes des 15 CA et 25 CC.

2.24. Classification à l'aide de machine.

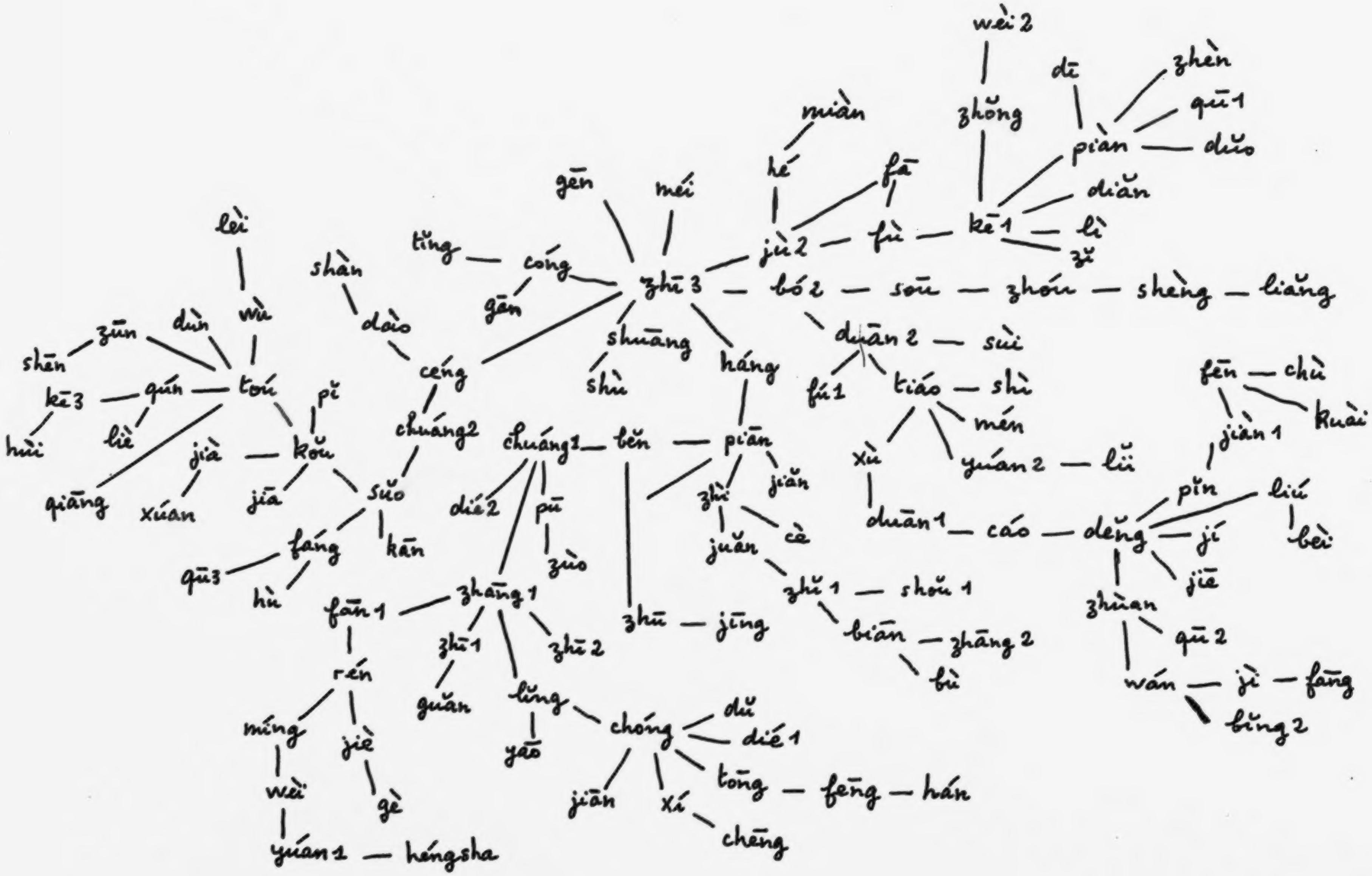
Les résultats donnés ci-dessus ont été obtenus à la main, au terme d'une étude ingrate et incomplète portant seulement sur les classes d'appartenance de 500 noms environ en CA et en CC. Afin d'obtenir des résultats exempts d'erreurs (ce que ne prétendent pas être ceux qui ont été présentés ci-dessus) et portant sur un échantillon important de noms (environ 3.000), on envisage une aide mécanique. Le but recherché est l'établissement correct des classifications de noms en CA et en CC : classes disjointes, classes sécantes, énumération exhaustive des éléments de chaque classe et de chaque zone d'intersection 2 à 2 et n à n. C'est seulement à l'issue de ce travail que l'on pourra avoir une idée précise de la classification des noms par les numérales en chinois, et de son évolution diachronique (en particulier de l'évolution de la numérale générale *gè* : le *gèhùa*). En outre, le programme sera assez général pour permettre une étude analogue des classifications nominales dans d'autres langues à particules numérales, comme le laotien, le vietnamien, le thai, le malais (Herrfurth), le japonais et le maya du Yucatan (cf. la 2ème partie de notre bibliographie). Ce n'est qu'alors que des comparaisons précises (et non plus vagues comme toutes celles qui ont été présentées jusqu'à présent) pourront être entreprises.

N.B. Les numéros de référence de l'index des numérales du CA donné en annexe se rapportent aux paragraphes de notre rapport détaillé (1967 b), reproduisant pour le CA l'ordre de présentation de Liú Shìrú (1965, p. 76 à 223).

Intersections entre classes nominales formées à l'aide des "numérales"
Chinois contemporain.


duī
cháng
chē
tái
jià
liǎng
shàn
dào
gè
bù
běn
piān
biān
dǐng
jiàn1
shēn
bǐ
tào
juǎn
zhuāng
wèi
míng
suǒ
chù
zhǒng
mén
zūn
shēng
zhū
kē2
kē1
fā
lì
zuò
jiā
chuáng2
céng
miàn
kuài
piàn
zhèn
duǒ
kǔn
zā
tiáo
wěi
duān2
gǔ
chuàn
zhāng
fú
bǎ
kǒu
tóu
gēn
guǎn
zhī1
hé
bāo
wǎn
bēi
píng
tǒng
dī
zhī3
qún
pī
bān
bāng
shuāng
fù
duì
guān
hú

Intersections entre classes nominales formées à l'aide des "numérales"
Chinois ancien (+220-600)


wèi 2
zhǒng
dī
zhèn
qū 1
piàn
dǔo
miàn
hé
fā
diǎn
gēn
méi
kē 1
lì
zǐ
tǐng
cóng
jù 2
fù
gǎn
zhī 3
bó 2
sōu
zhóu
shèng
liǎng
lèi
shàn
wù
dùn
zūn
dào
shuāng
duān 2
suì
shēn
tóu
céng
shù
háng
fú 1
tiáo
shì
fēn
chù
kē 3
qún
pǐ
mén
kuài
jiàn 1
huì
liè
jià
kǒu
chuáng 2
chuáng 1
běn
piān
xù
yuán 2
lǚ
pǐn
liú
zhì
jiǎn
qiāng
xúan
jiā
sǔo
dié 2
pū
cè
duān 1
cáo
děng
jí
bèi
juǎn
jiē
fáng
kān
zùo
zhǐ 1
shǒu 1
zhuan
qū 3
hù
fān 1
zhāng 1
zhū
jīng
biān
zhāng 2
qū 2
zhī 1
zhī 2
bù
wán
jì
fāng
rén
lǐng
bǐng 2
guǎn
chóng
dǔ
dié 1
míng
jiè
yáo
tōng
fēng
hán
wèi
gè
jiān
xí
chēng
yuán 1
héngsha

Index des numérales du chinois ancien

bèi ("cent chars")	輩	1.12 K
běn	本	1.12 B
biān	編	1.13 E
bǐng 1 ("manche")	柄	1.12 C
bǐng 2 ("pâté")	餅	1.13 H
bó ("jonque")	舶	1.13 I
bù	部	1.22 D
cáo	曹	1.12 K
cè	册	1.13 E
céng	層	1.12 J
chàng	唱	1.13 L
chēng	稱	1.22 C
chóng	重	1.12 J
chù	処	1.12 L
chuáng 1 ("lit")	牀	1.12 D
chuáng 2 ("drapeau")	幢	1.12 E
cóng	叢	1.22 A
dào	道	1.12 I
děng	等	1.12 K
dī	滴	1.12 F
diǎn	奌	1.12 G
dié 1 (trois champs)	疊	1.12 J
dié 2 (clef de la planche 90)	牒	1.12 J
dǔ	堵	1.13 C
duān 1 (clef li)	端	1.12 C
duān 2	段	1.12 H
dùn	頓	1.12 N

fú 1	("surface")	幅	1.12 I
dǔo		朵	1.13 K
fā		发	1.21 B
fān 1	(clef des plumes)	翻	1.12 N
fán		幡	1.13 J
fāng		方	1.13 B
fáng		房	1.13 A
fēn		分	1.12 H
fēng		封	1.13 F
fú 2	(clef de la main)	拂	1.12 I
fù		副	1.22 B
gān		竿	1.13 K
gè		个	1.11
gēn		根	1.12 B
gēng		更	1.13 L
gǔ		股	1.12 G
gǔan		管	1.13 D
hán		函	1.13 F
háng		行	1.22 E
hé 1		合	1.22 B
héngsha		恒沙	1.13 M
hù		户	1.22 D
hùi		会	1.13 M
jí		级	1.12 J
jì		剂	1.13 B
jiā		家	1.22 D
jià		架	1.21 B
jiān		间	1.12 H
jiǎn		简	1.13 E
jiàn 1		件	1.12 H
jiàn 2	("verrou")	楗	1.13 M
jiē		階	1.12 J
jié 1	("versifier", clef de l'homme)	偈	1.13 M

jiè	介	1.11
jīng	莖	1.12 B
jù 1 ("parole")	句	1.13 E
jù 2 ("préparer")	具	1.22 B
juǎn	卷	1.13 E
kān	堪	1.13 M
kē 1 (clef de la feuille)	顆	1.12 G
kē 3 (céréale et boisseau)	科	1.12 K
kè	刻	1.13 L
kǒng	孔	1.13 H
kǒu	口	1.12 A
kuài	块	1.12 G
lèi	类	1.12 K
lì	粒	1.12 G
liǎng	兩	1.13 I, 1.21 A
liè	列	1.22 E
lǐng	领	1.12 C
liú	流	1.12 K
lǚ	縷	1.12 F
lüǎn	臠	1.13 B
méi	枚	1.11
mén	门	1.12 K
miàn	面	1.12 I
míng	名	1.12 E
niǔ	鈕	1.13 G
pǐ	疋	1.13 I
piān	篇	1.13 E
piàn	片	1.12 G
pǐn	品	1.12 K
pò	璞	1.13 H
pū	鋪	1.12 I
qì	契	1.13 M

qiāng		腔	1.12 A
qū 1	("région")	区	1.12 L
qû 2	("tordre")	曲	1.13 E
qū 3	("corps")	軀	1.13 M
qún		群	1.22 A
rén		人	1.12 M
shàn		扇	1.13 C
shēn		身	1.13 M
shèng		乘	1.12 I
shì		事	1.12 M
shoǔ 1	("tête")	首	1.13 E
shù		樹	1.12 M
shuāng		雙	1.21 A
sōu		艘	1.13 I
sùi		穟	1.21 B
suǒ		所	1.12 L
tà		沓	1.22 B
tiáo		条	1.12 B
tǐng		梃	1.12 B
tōng		通	1.12 N
tóu		头	1.12 A
wán		丸	1.12 G
wèi 1	("rang")	位	1.13 A
wèi 2	("odeur")	味	1.13 B
wén		文	1.13 H
wù		物	1.12 M
xí		襲	1.22 C
xù		绪	1.12 C
xúan		懸	1.21 B
yāo		腰	1.12 I

yuán 1 (pour les personnes)	員	1.13 A
yuán 2 ("ourlet")	緣	1.13 M
zhāng 1 ("bander un arc")	張	1.12 I
zhāng 2 ("texte")	章	1.13 E
zhèn	陳	1.12 N
zhī 1 (clef du bois)	枝	1.12 B
zhī 2	支	1.12 B
zhī 3 ("oiseau")	隻	1.12 F
zhǐ 1 ("papier")	紙	1.12 M
zhǐ 2 ("doigt")	指	1.13 H
zhì	幟	1.13 E
zhǒng	種	1.12 K
zhóu	軸	1.12 C
zhū	株	1.12 B
zhuàn	轉	1.12 J
zǐ	子	1.12 G
zūn	尊	1.13 M
zùo	座	1.12 D

Liste des Numéraux du chinois contemporain

bǎ	把	fā	发
bān	班	fān 2 "retourner"	番
bāng	帮	fēng	封
bāo	包	fú 1 "surface"	幅
bēi	盃	fù	副
běn	本	gǎn	杆
bǐ	笔	gè	个
biān	编	gēn	根
bù	部	gǔ	股
céng	层	guàn	罐
chǎng	场	guǎn	管
chē	车	háng	行
chū	出	hào	号
chù	处	hé 2 "boîte"	盒
chuàn	串	hú	壶
chuáng 2 "drapeau"	幢	jiā	家
dài 1 "ceinture"	带	jià	架
dài 2 "sac"	袋	jiān	间
dào	道	jiàn 1 "part"	件
děng	等	jié 2 "noeud de bambou"	节
dī	滴	jù 1 "parole"	句
diǎn	点	juǎn	卷
dǐng	顶	kē 1 (clef de la feuille)	颗
duàn 2 "morceau"	段	kē 2 (clef du bois)	棵
duī	堆	kǒu	口
duì	对	kuài	块
dùn	顿	kǔn	捆
duǒ	朵	lì	粒
duò	垛	liàng	辆

liè		列
lǚ		缕
lún		轮
mén		门
miàn		面
míng		名
pán		盘
pī		批
pǐ		匹
piān		篇
piàn		片
píng		瓶
qún		群
shàn		扇
shēn		身
shēng		声
shǒu 1	"tête"	首
shǒu 2	"main"	手
shuāng		双
sōu		艘
suǒ		所
tái		台
tào		套
tiáo		条
tǐng		挺
tǒng		筒
tóu		头
wǎn		碗
wěi		尾
wèi 1		位
yǎn		眼
zā		扎

zhǎn		盏
zhāng 1	"bander un arc"	张
zhèn		阵
zhī 2		支
zhī 3		隻
zhǒng		种
zhū		株
zhuāng		桩
zūn		尊
zuò		座

Bibliographie

1. Analyses syntaxiques.(1)

Annear, S.S.- (1965) English and Mandarin Chinese : Definite and Indefinite Determiners and Modifying Clause Structures, POLA 11, (p 1-55), Ohio State Univ.

Coyaud, M.- (1967a) Les phrases hypothétiques en chinois contemporain (Problèmes de méthode). A paraître dans Hommage à A. Martinet. (A. Juilland ed.)

Elliot, D.E.- (1965) Interrogation in English and Mandarin Chinese, POLA 11 (p 56-117), Ohio State Univ.

Ertel, L.L., Garvin, P.L., Pang, F.L.L., Reitz, G.- (1965) Fulcrum technique for Chinese, English Machine Translation. The Bunker Ramo Corp., Canoga Park, California.

Žerebin, M.E.- (1961a) Struktura i harakter algoritma dlja mašinnogo analiza kitajskogo teksta. ITMiVT, Sbornik statej, p 173 sq.

(1961b) O razrešenii konversionnoj omonimii pri mašinnom analize kitajskogo teksta, ITMiVT, p 180 sq.

Hashimoto, A.Y.- (1965) A Condensed Account of Syntactic Analysis of Mandarin. POLA 10, (p 1-27), Ohio State Univ.

(1966) Embedding Structures in Mandarin, POLA 12, Ohio State Univ.

Li, K. P.- (1964) Automatic Recognition of Mandarin Monosyllables. POLA 9, Ohio State Univ.

Nikitina, T.N.- (1965) Obrabotka podležaščego pri MP s kitajskogo jazyka. NTI, 10- 1965 p 37-40

(1966) Grammatičeskaja neodnoznačnost' pri avtomatičeskom perevode s kitajskogo jazyka na russkij. NTI, 2- 1966 p 20-23

Pang, F.C.C.- (1966) A Structural Recognition Routine for the Noun Phrases of Standard Chinese, in Computation in Linguistics, (p 207-231), Garvin ed. Indiana Univ. Press.

Voronin, V.A.- (1961) Sistema nezavisimogo konfiguracionnogo analiza kitajskogo teksta pri mašinnom perevode, ITMiVT, p 135 sq.

Wang, S.Y.- (1963) Some Syntactic Rules in Mandarin, POLA 3, (p 32-83), Ohio State Univ.

Zemnov, A.A.- (1961a) Analiz smežnyx konstrukcij pri binarnom mašinnom perevode s kitajskogo jazyka na russkij. ITMiVT, p 349 sq.

(1961b) Analiz služebnyx slov pri binarnom mašinnom perevode s kitajskogo jazyka na russkij. ITMiVT, p 363 sq.

note (1) - Les références de [illegible], Voronin et Zemnov sont publiées dans les "Travaux de l'Institut de Mécanique de précision et de Technique du calcul, Académie des Sciences, 2e ed. Moscou, 1961 (ITMiVT).

2. Classification nominale.

Numérales du chinois.

Coyaud, M.- (1967b) Classification nominale en chinois :les particules numérales. Rap/P.2/SAD Polycopié, CNRS,SAD, Paris.

Dragunov.- (1958) Xiàndài hànyǔ yǔfǎ yánjiū, Pékin.

Herrfurth, H.- (1964) Die Klassifikationsaffixe im Chinesischen, Beiträge zum Problem des Wortes im Chinesischen II, p 1–43. Akademie Verlag, Berlin,

Huáng Zàijūn.- (1964) Cóng jiǎwén, jīnwén liàngcíde yīngyòng, kǎochá hànyǔ liàngcíde qǐyúan yǔ fāzhǎn. Zhōnggúo yǔwén nº 6, p 432–441.

Jaxontov, S.E.- (1965) Drevnekitajskij jazyk. ANSSSR, Moscou.

Líng Quánfāng.- (1955) Yòng yǔ rénde gè tǐ liàngcí. Yǔyán zhīshí, nº 6.

Líu Shìrú.- (1965) Wèi Jìn Nǎn Běi cháo liàngcíde yánjiū. Zhōnghúa shūjú. Pékin.

Líu Shìshēng.- (1957) Tán liàngcí.Yǔyán xúexí, nº 10.

Wǎng Lì (Wang Liaoyi).- (1955) Yǒuguan rén wù hé xíngwéide xūcí. Yǔ wén xúexí cóngshū, Zhōnggúo qīngnían chūbǎnshè, Pékin.

Zhōng Běnkāng.- (1959) Shìwù hé liàngcí. Yǔyán zhīshí, nº 8.

Voronin, V.A.- (1961) Sistema nezavisimogo konfiguracionnogo analiza kitajskogo teksta pri mašinnom perevode, ITMiVT, p 135 sq.

Wang, S.Y.- (1963) Some Syntactic Rules in Mandarin, POLA 3, (p 32-53), Ohio State Univ.

Zvonov, A.A.- (1961a) Analiz ramočnyx konstrukeij pri binarnom mašinnom perevode s kitajskogo jazyka na russkij. ITMiVT, p 349 sq.

(1961b) Analiz služebnyx slov pri binarnom masinnom perevode s kitajskogo jazyka na russkij. ITMiVT, p 360 sq.

note (1) - Les références de Žerebin, Voronin et Zvonov sont publiées dans les "Travaux de l'Institut de Mécanique de précision et de Technique du calcul, Académie des Sciences, 2e ed. Moscou, 1961 (ITMiVT).

Numérales d'autres langues.

Deloustal, R.- (1914) Des déterminatifs en annamite, BEFO, XIV, 5.

Emenau, M.B.- (1951) Studies in Vietnamese Grammar. Univ. of Calif. Press, Berkeley.

Haas, M.R.- (1942) The use of numeral classifiers in Thai, Language, XVIII, p 201 sq.

Haguenhauer, Ch.- (1951) Morphologie du japonais moderne, Klincksieck, Paris.

Hospitalier, J.J.- (1937) Grammaire laotienne, Imprimerie Nationale, Paris.

Koutsoudas, A.- (1966) Writing Transformational Grammars. Mc Graw-Hill, New-York.

Naganuma, N.- (1951) Basic Japanese Course, Tokyo.

An evaluation of the usefulness of machine translations produced at the National Physical Laboratory, Teddington, with a summary of the translation methods.

J. McDaniel, W.L. Price, A.J.M. Szanser, D.M. Yates

Introduction

The machine translation project at the National Physical Laboratory (NPL) has been terminated. It has always had as its prime aim a demonstration of the practicability of translation by computer of Russian scientific texts into English. In order to test how far this aim has been fulfilled and further, to provide evidence to guide a potential agency interested in giving a machine translation service, we have carried out an evaluation experiment on our translations, the conditions of which as far as possible emulated those of a translations service.

The results of this experiment are presented in this paper, together with a summary of the translation methods used. The paper as a whole will thus give an independent presentation of "what methods produced what results". For a comprehensive account of the NPL translation techniques, see reference 1.

Evaluation of Translations

We have been concerned with the translation of scientific Russian texts only. In considering how we might evaluate the results of our work, the context of use of scientific translations imposed two main constraints. Thus, firstly, in the vast majority of cases we would expect readers of translations to be themselves experts in the subject matter of the material translated, i.e. they would be reading the translations because these reflect their main professional responsibilities. We may then expect that the inherent background knowledge of such readers will ensure a high impetus to their comprehension of translations and help them through syntactic awkwardnesses and multiple-meaning choices. We would also expect that only a small percentage of these readers would have any competence in Russian. Secondly, the items of translation being read by the above typical readers will normally be whole information units (journal article, chapter of book, abstract, review, &c.), and they will have the freedom to ignore unimportant sections of such units and to use sentence or paragraph context (or even remoter references) to help elucidate obscure sections. More specifically, a particular sentence may be poorly translated, but because the reader can see that this is not an important sentence or because the context of (hopefully, better-translated) neighbouring sentences clarifies its meaning, that sentence may not affect at all an adequate comprehension of the whole.

Both these constraints are reflected in our evaluation experiment. We ensured that our evaluators were expert in the field of the material they were evaluating, and also that they commented on the adequacy of an information unit as a whole, not on individual sentences.

We have included in this paper (FIG. 2(A)) a short passage from one of the evaluated translations, as the full translation is inappropriate for this printed version. However, the full translation will be available for inspection at the presentation of the paper, or the full translation of another paper can be examined in reference 1.

The evaluation experiment

In order to fulfil the first constraint above, we invited practising scientists to send in Russian papers, reflecting their professional speciality and preferably in the fields of general physics, electronics, or electrical engineering. Some papers resulted from direct invitation, others resulted from an open invitation published in our house journal, "NPL Quarterly". We undertook to send them the machine translations of their papers in return for their comments on how useful the results were. We also obtained second opinions from other specialists in the subjects concerned.

These evaluators were therefore as far as possible typical of the "customers" of a production MT service; in particular they had a personal interest in the subject matter and usually little if any knowledge of Russian.

In all 44 papers were received in response to our invitation; of these 28 were translated in full[1]. Seven of these were disregarded for various reasons[2], and the remaining 21 were included in the evaluation.

38 comments were received on 19 of these 21 papers. Of these two were rejected for vagueness, and three brief comments from one group were treated as one, so in all the experiment produced 34 comments on 19 papers.

---

[1] The other 16 are accounted for as follows: 1 was on a remote subject; 2 were deferred since we had already translated three papers for the same 'customer'; 4 were withdrawn; 3 were translated only in part; and 6 were not reached by the date our computer was scrapped.

[2] 3 were on inappropriate subjects; 2 were translated only by an earlier version of the programs; and 2 were translated too late for inclusion.

We had decided to give our evaluators a free hand in discussing the usefulness to them of translations of this quality. This meant that a scale had to be devised by which their comments could then be graded by us. A scale recently published in the U.S.A. (reference 2) was considered but not adopted since we felt that for our purposes more space should be given to the middle range of the scale. The following wording was adopted:

8 Fully adequate. Meaning immediately clear, even though not always conventionally expressed.

7

6 Mostly very good. A few sentences obscure, so that something essential may be lost, but normally clear enough.

5

4 Fair. Takes a good deal of time to extract meaning and even then there is no great confidence in it, resulting in a partial understanding.

3

2 Poor. Could only be useful to someone prepared to struggle hard, and even he would often be disappointed.

1

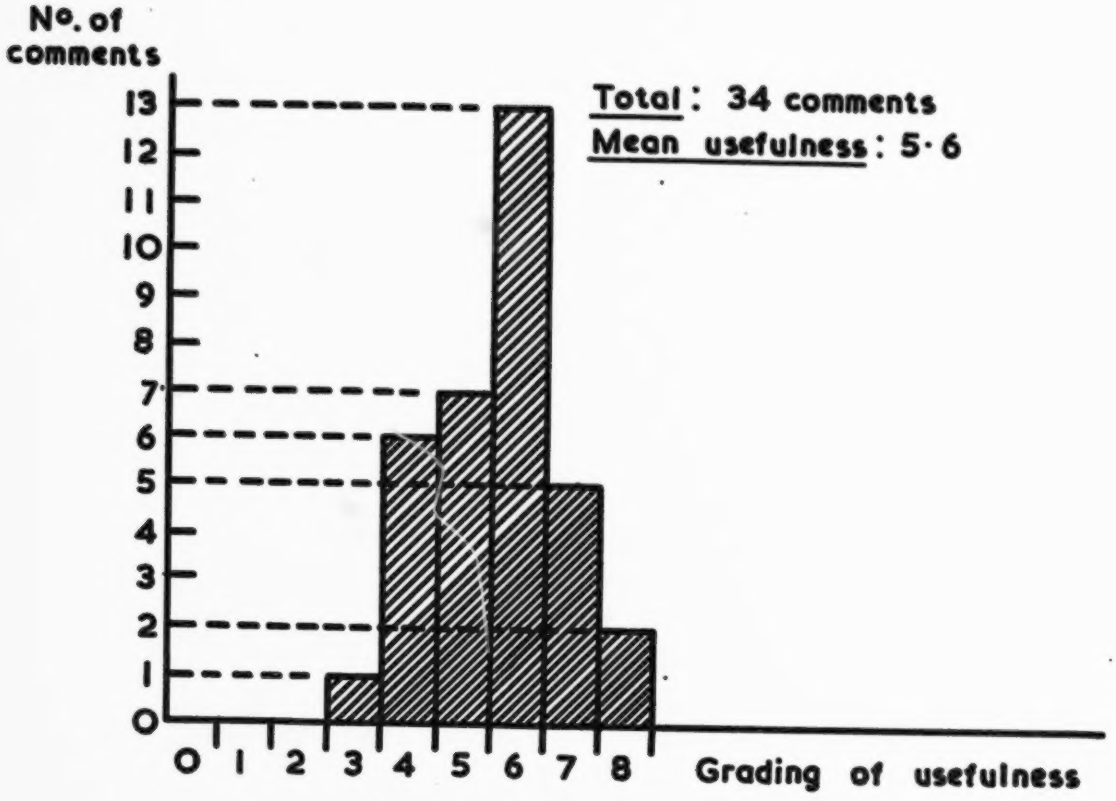
0 Useless. Although some semblance of meaning may appear occasionally, it would never be worth the trouble of finding it.

The wording of this scale is not derived on any scientific basis, but it has proved useful in practice, since when four of us came to grade the comments by it independently, there was a good agreement between our markings. Our four individual ratings for each comment were reduced to a single rating (normally the mean) after discussion. The range of scores is shown in FIG. 1; the mean score is 5.6.

The spread is no doubt due to a real variation in the quality of the translations combined with the prejudices and degrees of patience of the evaluators. The lowest scores thus come from impatient professional translators dealing with a poorer-than-average text, while the highest ones are perhaps over-enthusiastic supporters dealing with a better-than-average text.

The consensus though, is that there is a real demand for translations of this quality, and this result provides, we feel, ample justification for mounting a broader evaluation exercise, over a wider range of potential readers of such translations, to strengthen, if possible, this verdict and make it possible to

FIG. 1 Assessment of usefulness of N.P.L. MT output.



decide on the viability of a production machine translation service based on our system.

Evaluators' criticisms

Apart from the opinions as to the general usefulness of translation, evaluators' comments contained many particular points of criticism which deserve discussion. We are able to comment ourselves on some of these points from the position of having done considerable development work, just short of full implementation, on techniques designed to overcome the particular translation faults. Full details of this further work are given in reference 1, and specific points of reference are given below.

Most of these criticisms can be classified into three groups, concerning respectively: (i) the English equivalents offered, (ii) the syntactic resolution and (iii) the word order.

A frequent criticism concerned missing or inappropriate equivalents. In addition to fully justified remarks of this kind there were also cases in which the meaning proposed, or preferred by the reader, was uncommon. Its absence from the dictionary was the result of a preferential choice having been made, a compromise between completeness and simplicity. The other alternative, including all possible equivalents, would of course drastically impair readability. The particular solution is often very difficult and can only be achieved to a satisfactory degree after long experience.

In other cases there is no obvious preference and the problem is further aggravated by the very high frequency of occurrence of the word. Here belong some special classes, for example all prepositions and some very common words such as и , а , and что . Prepositions can and should be resolved by considering them together with either the governing word or the governed complement (nominal or otherwise)[1]. (For example, увеличить.. на.., 'to increase .... by ....'). For the awkward common words specific syntactic sub-routines should be devised. In practically all cases the solution is unique (see reference 1).

Only two evaluators complained about the necessity of selection among two or three equivalents. This is a matter of preference, but it seems to us that for a bona fide reader an additional possibility of meaning (if it is not carried too far) is more an asset than a disadvantage, even if it impairs to some extent smooth reading[1]. Until a semantic analysis can be achieved, multiple equivalents are bound to stay in MT.

---

[1] On the lines already used for the recognition of idioms, expanded to include non-adjacent words; see below in the summary of methods.

A minor point, but nevertheless worth attention, was to the effect that when multiple equivalents followed each other, the difficulty in understanding increased out of proportion. For example, случается при appears as: 'occurs in ' when the
results with
actual meaning is often 'results in'. This was undoubtedly a real problem, which could perhaps be helped by using a longer space between sets of multiple equivalents in the output.

Complaints concerning un-idiomatic translations (e.g. 'period of work' instead of 'life-time') would be allayed by more work spent on our idiom list, which contained only about 540 items, whereas 1,500 would be a more realistic figure.

Complaints about inadequate syntactic analysis, leading to obscurities, ambiguities, and wrong resolutions, would have been considerably reduced by a full implementation of the syntactic routines described in reference 1. One of the minor but annoying ambiguities, which had been resolved theoretically, but only partially implemented, was that of adverb/short adjective. Order of clause components was a frequent subject of criticism; of course they can be re-arranged according to the English usage only after a complete analysis has been made.

Among other things criticized was an inadequate treatment of abbreviations and abbreviated units, some of which were covered by dictionary entries, while others were not, and this led to some misunderstandings. Obviously this again is a matter for a more complete dictionary[2]. The most difficult case is "nonce" abbreviations (we met, for instance, нейтр. for нейтронный and produced 'non-itr.', which helped no one!) Here we see no prospect of a solution.

Our "anglicizing" routine was criticized (while appreciating the general idea) for unorthodox transliteration, which made it more difficult to identify the word in a standard dictionary, if necessary[3]. A partial solution may be to exclude certain word classes, e.g. acronymic abbreviations, which are obviously not suitable objects for the routine (they can be automatically recognized as clusters of capital letters). Also, in our prefix-recognizing routine there is an inherent danger that a "not-in-dictionary" word may have a part of the stem identical with an accepted prefix. This applies in particular to short prefixes, like не- , in the above example of нейтр.. There is no general way of dealing with such words. The best solution, in respect of both routines, seems to be, however, to include in the output both the original (in Cyrillic, if possible) and the synthetic equivalent for all "not-in-dictionary" words.

---

[1] Much can be said on this point. Readers, no doubt, will realise how a velvet smoothness of translation may hide many a grievous fault.

[2] With a few exceptions, however. Thus 'в' may be very troublesome, as regards the choice between the preposition and the abbreviated unit ("volt"), without a special syntactic subroutine.

[3] This criticism clearly implied some knowledge of Russian.

A few comments contained bouquets rather than brickbats. One evaluator commented that the translation became easier to read as he got used to the unusual 'style'; and another found an instance where a slip in the published human translation had reversed the intended meaning; our version of the passage, while not perfect by any means, was certainly not misleading in this way.

Finally, several evaluators commented that machine translations would need to show advantages in cost and speed over human translations in order for them to be attractive as well as acceptable, and these are indeed criteria that we would ourselves put forward without fear of contradiction. We have not included a study of cost and speed within this evaluation experiment, as we do not have the market data to prepare a translation service specification that we could then refer such a study to. However it is evident that our machine equivalent of the human translator i.e. input punching, machine translation and output printing (with no human post-editor) will show a clear advantage on both these points. It would be essential to fit this component, though, into an overall translation system which was specified carefully to fit the translation market.

In FIG. 2(A) is shown a facsimile of a short passage of our machine translation into English of a Russian text on electric furnaces, completely non-post-edited. The vertical lists of two or three words are to be read as alternative English correspondents for the Russian word in that position. FIG 2(B) is a facsimile of the original Russian text.

A summary of the translation methods

Text Preparation and Dictionary Look-up

The dictionary used in the NPL machine translation system was developed from an early version of the Harvard Russian-English computer dictionary. Our dictionary contains about 18,000 entries (with additional cross-reference entries) covering the fields of electronics and electrical engineering.

We chose to organize the dictionary on a stem and suffix

FIG. 2(A) English machine translation

```
Metal melted into furnace(s) is possible to present in the form of continuous
             in

block    , but then to cut out from it elementary cube of any dimension and  to
assembly   and         engrave                                size      also

define    its resistance.
determine

Having replaced elementary cube of melted metal by unit of electrical circuit of
                                                   node
                                                   knot

model, is possible to reveal distribution of current in it and  , having
                                                           also

modelled thus all bath of furnace, is possible to recognize character of
                                                  learn

distribution of current in melted metal.

Constructional grid model represents geometrically similar volume of bath in
constructive

significantly decreased scale.
```

FIG. 2(B) Russian original text.

распределения электрического тока ванне расплавленного металла

(андидат техн. наук, доц. А. И. ЛЕУШИН
шевский индустриальный институт им. Куйбышева

ектрического то-
ической печи, как
ольшое теоретиче-
Поэтому вопросы,
сона (Л. 1), тре-
изучения. Особую
вание распределе-
ым металлом. Вы-
ванне печи настоя-
мпленного реше-
о перемешивания
еление характера
дать более рацио-
/ и конструкцию
числе и размеще-
х печах, правиль-
и электромагнит-
в. В настоящей
ся применительно

авленном металле
от формы ванны,
ружения электро-
лектрический ток
характеризуется
уравнением Ла-

=0

ды;
кого поля.
тока через мас-
зается действием
ными точкам на
ть вектор плотно-

ктрическая цепь

ления тока в сплошных проводящих средах применяются три вида моделей: проводящие пластины, или листы, электролитические ванны и решетки, или сетки. Использование решеток, или сеток, из сопротивлений имеет бесспорное преимущество по сравнению с остальными способами, так как позволяет непосредственно исследовать распределение тока в модели.

Для осуществления подобия физических процессов объекта и модели необходим правильный выбор критериев подобия. Необходимые и достаточные условия подобия физических явлений устанавливаются третьей теоремой подобия, доказанной еще в 1930 г. М. В. Кирпичевым.

Общий критерий подобия

$$k = l\sqrt{\omega\gamma\mu},$$

где l — линейные размеры;
ω — угловая частота;
μ — магнитная проницаемость;
γ — удельная проводимость.

При равенстве ω и μ объекта и модели наиболее важными критериями подобия являются размеры и проводимость материала сетки модели.

Расплавленный в печи металл можно представить в виде сплошного блока, а затем вырезать из него элементарный куб любого размера и определить его сопротивление. Заменив элементарный куб расплавленного металла узлом электрической цепи модели, можно выявить распределение тока в нем и, смоделировав таким образом всю ванну печи, можно узнать характер распределения тока в расплавленном металле.

Конструктивно сеточная модель представляет собой геометрически подобный объем ванны в значительно уменьшенном масштабе.

Сопротивление элементарных кубиков металла имитируется сопротивлением соединительных проводов — шага ячеек модели. Шаг сетки зависит от геометрических размеров объекта и модели и, следовательно, от общего количества ячеек модели. Точность моделирования будет тем выше, чем больше число ячеек. Однако слишком большое число ячеек ухудшает условия измерения и увеличивает габариты модели и материальные затраты на нее.

basis, in which each entry contains a Russian stem together with a coded list of suffixes which can combine with the stem. This gave far fewer entries than would have been found in a full-form dictionary covering the same words. Each entry contains grammatical data and English equivalents of the Russian.

The stem and suffix organisation demanded that we create a system of splitting Russian words consistently into stem and suffix, fully described in Davies & Day, (1961). The split is made at the point determined by the maximum number of letters which together form a Russian suffix or string of suffixes. The maximum split technique sometimes causes too many letters to be treated as part of the suffix, in other words, the split is made too early in the word. Such words are provided with a cross-reference dictionary entry which directs the search to an entry in which the full information for the word is contained.

The dictionary is recorded on two reels of magnetic tape, the entries being arranged in alphabetical order. Time of consultation of the full dictionary is from 12 minutes upwards, depending on the number of entries being sought.

A text for translation is first punched on cards by an operator who recognises Cyrillic characters, though she cannot read Russian. Symbols, punctuation marks and Cyrillic characters are represented by one card column per character. Provision is made for indicating a space to be left in the text where an equation or group of symbols occurs. These will be inserted in the translation by hand. The cards are treated as a continuous medium, card boundaries being ignored. By this means quite a long paper can be encoded on a relatively small number of punched cards.

The text, now on cards, is fed into the computer. The first computing process gives a serial number to each text word and then splits the word into stem and suffix. When all text words have been subjected to this process, they are then sorted into alphabetical order. This is essential for optimum speed of look-up in our serially organised dictionary.

The next programme in the translation sequence, the look-up programme, scans simultaneously through the dictionary and the sorted text, seeking dictionary entries corresponding to the text words. The programme allows for the occurrence of stem homographs and for the correct handling of cross-reference entries. The output of the programme (which we call, following Harvard, the augmented text) consists of the text words each with the relevant dictionary entries appended.

Having obtained a set of augmented text entries, the translation sequence then sorts these back to text order, using the text serial number originally allocated to each text word.

The result of this series of operations is a text in the original order, with dictionary entries appended to all but a few of the items. Symbols and punctuation marks do not, of course, have corresponding dictionary entries, and there may be words in the text which are not represented in the computer dictionary. The latter are given special treatment in the syntactic routines and translation output.

Provision is made in the dictionary for the representation of idioms, using a method analogous to that used in an ordinary dictionary. A "key word" is chosen in the idiom (normally the least frequently occurring word), the idiom being represented in the dictionary entry of the key word. The representation includes a list of the component words of the idiom, using which the presence of an idiomatic text word sequence can be detected before attempting any syntactic operations on the augmented text. The dictionary entry including the idiom contains the preferred English equivalent. The dictionary includes coding for 540 idioms.

Words not represented in the dictionary are given special treatment, as mentioned above. All text words which commence with one of a set of 137 Russian prefixes are looked up both with and without prefix. If the prefixed form does not occur in the dictionary, but the unprefixed form is found, then the entry for the unprefixed form is included in the augmented text, coupled with an English rendering of the Russian prefix. Despite this provision, some text words will not intersect with the dictionary. For these an attempt is made to determine part of speech, case, number, etc., by an inspection of grammatical and derivational suffixes. In the translation output the stem of the not-in-dictionary word is transliterated, aiming to anglicize as far as possible the original word. A derivational suffix is given its English equivalent in the output rendering; any prefix that was recognised is also given its English rendering.

From an augmented text produced by the foregoing procedures it would be a simple mechanical process to achieve a word-for-word "translation". We felt this was not worthwhile, as the application of relatively simple rules of grammar and syntax greatly enhance intelligibility of such a product.

Russian Analysis Algorithm

In the first place we designed and implemented a system of noun blocking and a simple predicate analysis. The results obtained were not by any means ideal, but we were encouraged to extend and refine our syntactic processes. In our first attempt the functions of Russian analysis and English synthesis were closely interwoven. As our syntactic procedures were extended to cover more features it became evident that it was essential to separate the functions of analysis and synthesis. In order to make this possible the linguistic model, described in Yates (this conference) was developed. The model permits the analysis routines to express the Russian syntax as far as necessary and facilitates a transformation to the corresponding English sentence structure.

The analysis routines operate in a succession of passes through each sentence, defined by major punctuation mark boundaries (full stop, question mark and semi-colon).

The functions of the successive passes are as follows:

1. A preliminary pass which establishes from the augmented text the terminal element for each discrete member of the sentence. Punctuation marks are indicated in the elements for preceding or following sentence items, according to a set of formal rules.

2. A pass whose prime concern is the determination of nominal structures, i.e. nouns and words with which they are closely connected, such as adjectives or prepositions.

3. A pass which establishes links between adjacent nominal structures; the linked elements include genitive qualifiers and prepositional group qualifiers.

4. A pass which searches for potential coordinating conjunctions and examines the sentence elements or structures separated by such conjunctions, setting up coordinate groups where appropriate.

5. A pass which creates simple predicate structures, searching for words with a verb role and then locating adjacent sentence elements or structures acting as verb adjuncts.

6. A pass whose function is to examine the role of some of the more "difficult" words such as the verb быть and its inflected forms, and the personal/possessive pronouns его, ее and их.

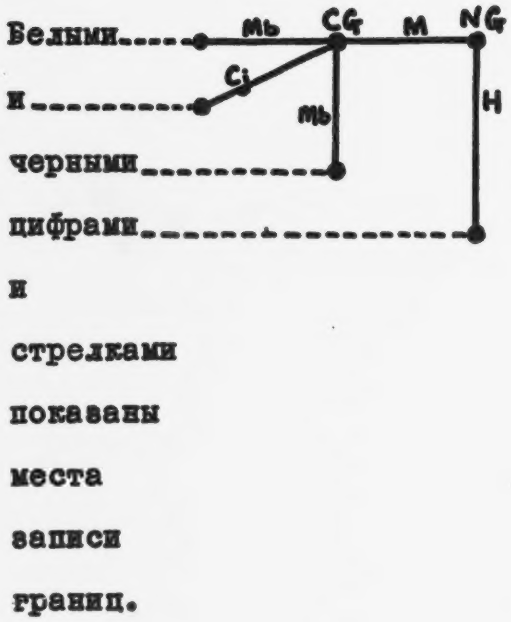
A full description of these analysis routines is given in reference 1. In the present paper we shall take a Russian sentence and note the effect of each analysis pass on it.

The Russian sentence reads:

Белыми и черными цифрами и стрелками показаны места записи границ.

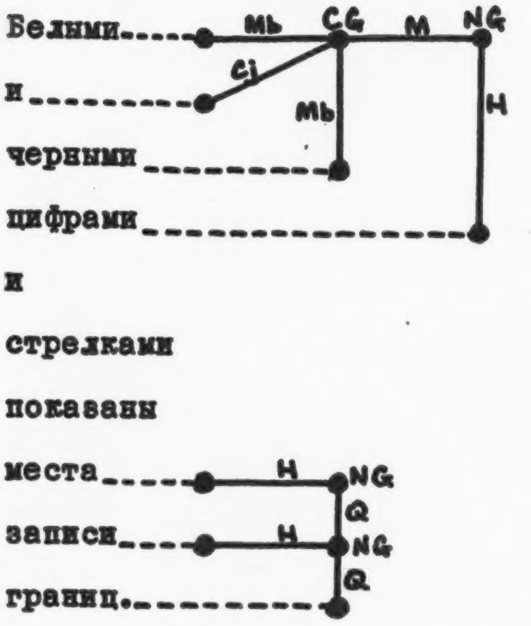
The first analysis pass is not of particular interest in the present context. Suffice it to say that a system of reference addresses is set up which permits the scanning of the sentence whilst its structure is in an incomplete state.

The state of the sentence diagram after the second pass has been completed is:-


Белыми
и
черными
цифрами
и
стрелками
показаны
места
записи
границ.
Mb
CG
M
NG
Ci
Mb
H

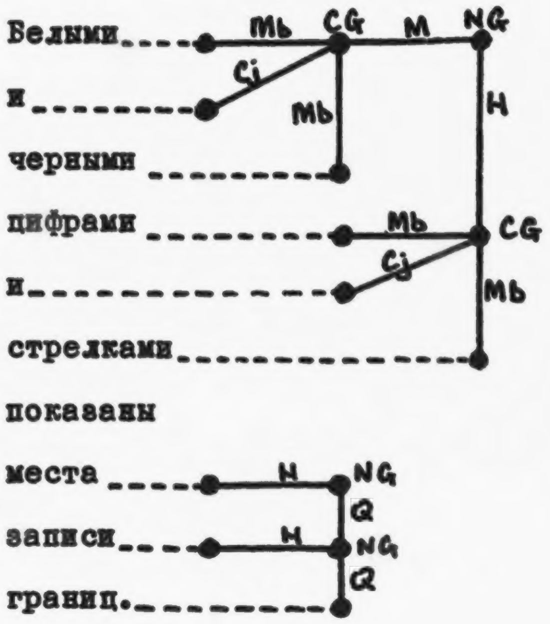
One noun group has been formed, of which the modifier is a coordinate group of adjectives. Each adjective is marked as a member in the coordinate group, which itself assumes the properties of an adjective.

The third analysis pass has the function of creating genitive and prepositional links. Only the former are concerned in our sample sentence:-


Белыми
и
черными
цифрами
и
стрелками
показаны
места
записи
границ.
Mb
CG
M
NG
Ci
Mb
H
H
NG
Q
H
NG
Q

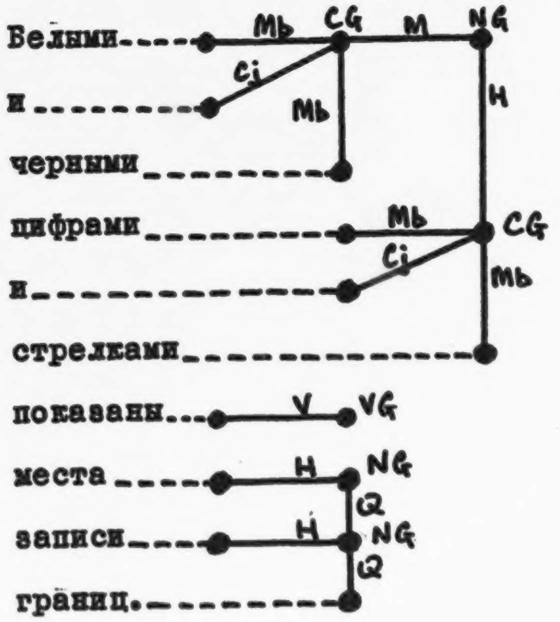
Were there any prepositional groups following nouns, then the prepositional groups would also be linked in as qualifiers.

The second analysis pass ignored all conjunctions which did not occur explicitly within simple noun groups (i.e. groups with a single noun as head). The fourth pass, however, seeks to join to existing noun groups any other nouns linked by coordinating conjunctions:-


Белыми
и
черными
цифрами
и
стрелками
показаны
места
записи
границ.
Mb
CG
M
NG
Cj
Mb
H
Mb
CG
Cj
Mb
H
NG
Q
H
NG
Q

In addition the pass groups together in coordinate groups any similar words joined by coordinating conjunctions, whatever their part of speech. Intervening punctuation prevents the formation of coordinate groups. The coordinate group, when formed, is given the grammatical significance of its component parts.

The fifth analysis pass has little effect on our sample sentence. The plural, short form participle is the single "verb" member of its verb group:-

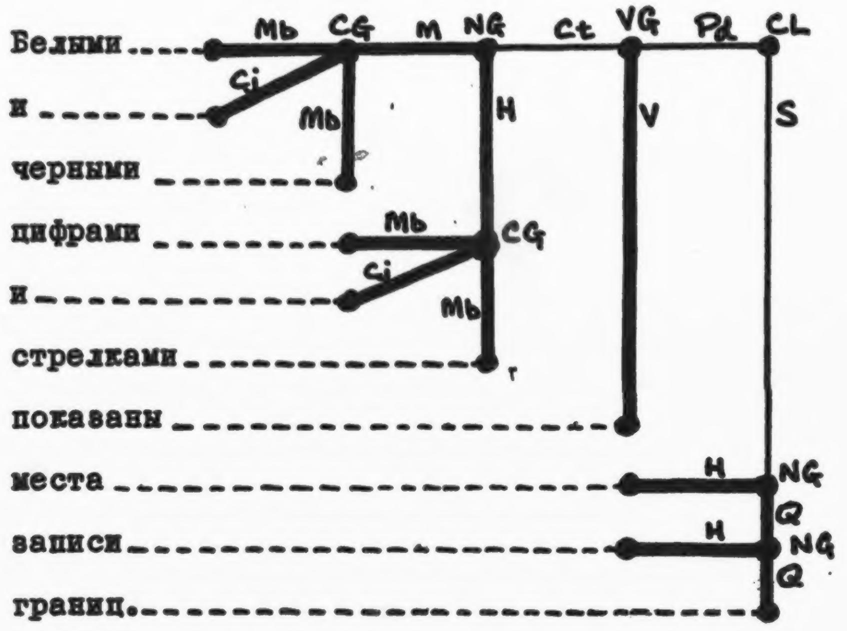


Were there adjacent adverbs or prepositional groups, these would be included in the verb group with the role of adjunct. The fifth pass also has provision for negative and conditional predicate structures.

The function of the sixth pass is to try and resolve the roles of certain more "difficult" words. (No instances occur in the sample sentence). For example, if one of the ambiguous personal/possessive pronouns is encountered, a check is made to see whether the following sentence element is nominal. If it is, then the pronoun is joined in the element as a modifier, and the pronoun is treated as possessive. Forms of the verb быть which were not covered by the provisions of pass five, are also included in the sixth pass.

Having completed the sixth pass, no further analysis of the Russian sentence is undertaken. The sentence structure delineated by the analysis passes is not complete, since no attempt is made to set up a clause structure. However, in order to facilitate the task of the synthesis procedure, all the separate group structures (and any remaining separate elements) are arbitrarily connected together in one or more higher groups, presenting the appearance of a unified whole to the synthesis stage.

On the other hand if further analysis passes were applied, particularly with reference to clause delimitation (for example, see Appendix 1 of reference 1), then the sample sentence would appear as:-



Thick lines indicate those connections which our analysis routines have created, and the thin lines indicate those which would be created by additional routines.

The translation sequence is completed by an English synthesis process. This determines re-orderings, insertions, inflections and selections of English equivalents, and, finally, the format of the printed output, produced by the computer on paper tape and printed on a flexowriter. This process is described in the companion paper, which also includes an account of the descriptive model mentioned above.

References

1. McDANIEL, J., DAY, A.M., PRICE, W.L., SZANSER, A.J., WHELAN, S. and YATES, D.M. "Translation of Russian scientific texts into English by computer — a final report". National Physical Laboratory, Autonomics Division report 35, June 1967.

2. National Academy of Sciences/National Research Council, "Language and machines; computers in translation and linguistics". 1966.

3. DAVIES, D.W. and DAY, A.M. "A technique for consistent splitting of Russian words". Proc. Intl. Conf. on Machine Translation of Languages and Applied Language Analysis, H.M. Stationery Office, 1962, 1, 343-362.

4. YATES, D.M. "A computer model for Russian grammatical description, and a method of English synthesis in machine translation". This Conference.

The work described above was carried out at the National Physical Laboratory.

3

Programming of Reversible Systems in Computational Linguistics

Gerhard Engelien, Forschungsgruppe LIMAS, Bonn

In my paper I shall report on some aspects of programming reversible systems, in particular the special problems involved with programming non-numerical systems.

Dr. Alfred Hoppe of Forschungsgruppe LIMAS has developed a procedure for a reversible data flow (Figure 1). In the matrix three different quantities are connected together. For analysis the grammatical description of an inflected form is determined by combining the inflection class with the ending. For example the German verb gehen is composed of the stem geh and the ending en. The lexicon tells us that geh belongs to inflection class B. By combining the channels of inflection class B and the ending en we are able to determine the grammatical description of the verb (which is in this case ambiguous). Each of these groups contains a series of binary variables whose value is either zero or one. The synthesis process unites the channels with the grammatical variables, the result being the appropriate ending.

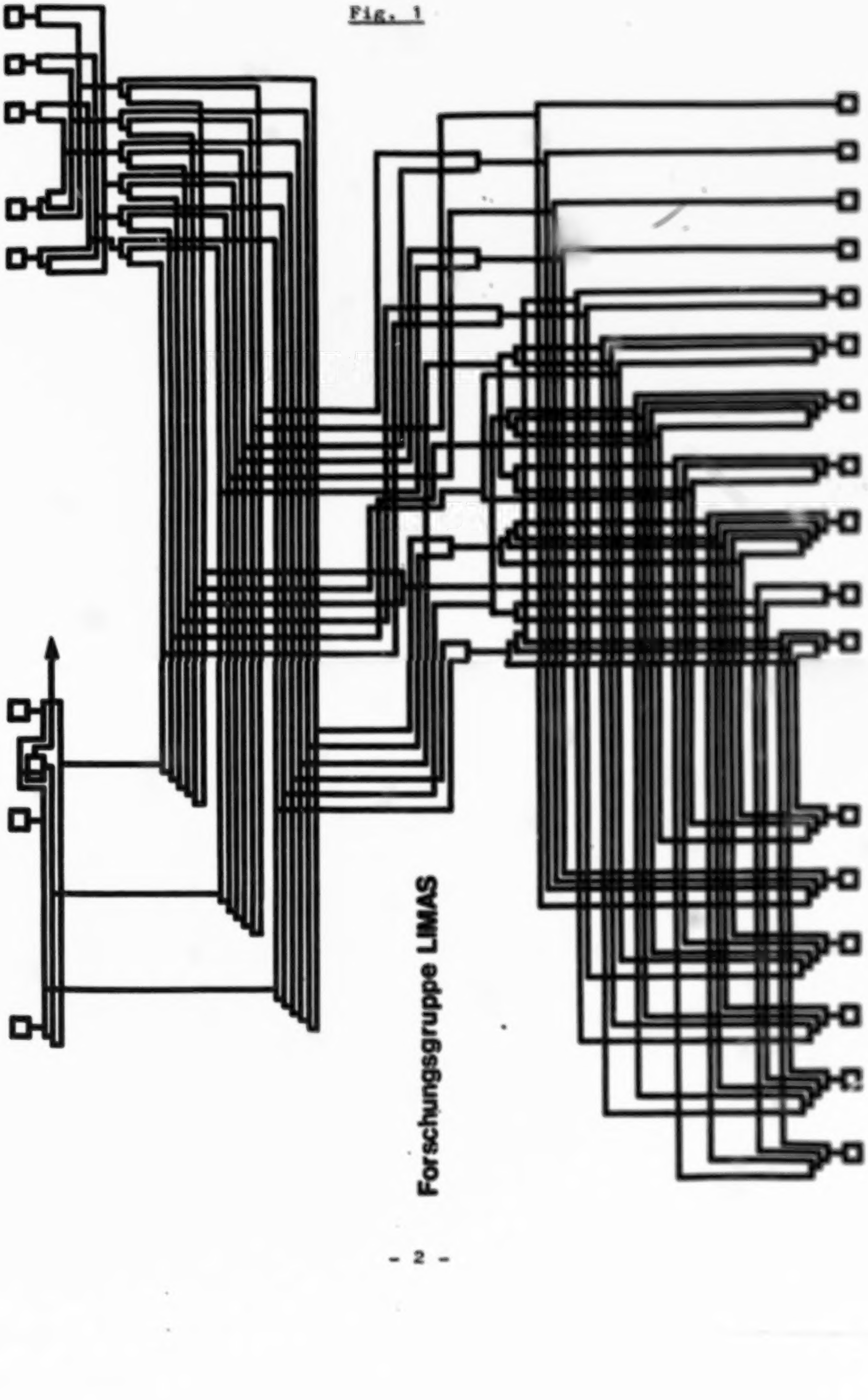
The matrices contain only "and" and "or" gates and an element which can block the flow in one direction, as certain linguistic forms will be analyzed but not synthesized. Technically realized, the matrix displays an example of parallel processing which is both complex as well as rapid.

For the present time the LIMAS system will be programmed for conventional computers, which operate more or less sequentially. Thus programming such a matrix involves simulating this parallel data flow on a sequentially operating machine. One must, however, take into consideration that this will result in an increase of running time. But for purposes of testing whether or not the end data have been correctly combined to deliver the final result this time increase is irrelevant.

Besides the parallel processing one must also consider reversibility. We have discovered that reversibility has not made programming more difficult; on the contrary, it has facilitated the program debugging process.

At first the matrices will be programmed to flow only in the direction of synthesis. For analysis one would have to write every possible form of a large number of words on punch cards in order to test all the variations. This, of course, would require an immense amount of work. It is a great deal simpler to let the computer print out the inflected forms, and then check the output to see if they are correct. The feature of reversibility thus makes it possible to test both directions, i.e. analysis and synthesis, at the same time.

Fig. 1


Forschungsgruppe LIMAS

I would like to discuss briefly various procedures which facilitate the simulation process. All descriptive categories such as case, number, person, etc. are coordinated with the various endings. The problem is therefore to compute the combined values of certain binary variables from the combined values of certain other binary variables. List processing is probably the best suited method for this type of problem.

Figure 2

	Ending	Inflection Class	Grammatical Information
	. X ..	. X . .	. X . . . . X . . . .
geh/	en	B	1-pl 3-pl etc.

The variables assume their meaning according to the arrangement in the rows and columns. It is obvious that each line of such a list can be stored in one or more machine words.

It is necessary to have a program which will encode endings, inflection classes, and grammatical information in bit-combinations, and a search program which can locate the respective bit-combinations in the list. For analysis the suffix bit pattern and inflection class bit pattern are given, and the list is searched for the line in which they are stored. This line also contains the grammatical bit pattern, the object of the search. The results can be printed out with the help of a decoding program, or else they can remain encoded and be processed at a higher level in the system. The synthesis process is similar except that the grammatical bit pattern is given in place of the suffix bit pattern. The suffix bit pattern is then the object of the search.

The list method has the advantage that necessary expansions or changes can easily be carried out during the developing stages without affecting the actual program.

Another programming possibility results from a formula-like representation. All variables of a line in the list are to be consecutively indexed.

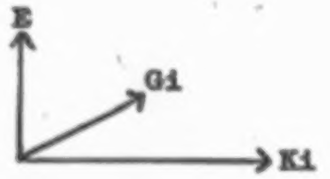
$$(a_1, a_2, \ldots, a_n)$$

With the aid of logical functions (e.g. the Boolean full form) it is possible to represent the process in the following way:

$$a_i \Leftarrow F_i (a_1, a_2, \ldots, a_n) \qquad i = 1,n$$

A search procedure is no longer necessary. Instead, as many logical equations will be computed as there are variables. Of course the variables will have to remain stored until the final result is available. It is possible to obtain these formula-like representations automatically from the lists, and to automatically minimize the obtained formulas, i.e. to establish the smallest number of binary operations.

The lists can also be stored in a three-dimensional binary matrix.



$M_{k,g,e}$,

Such matrices can be processed by formulas of the following type:

$$G_g \Longleftarrow E_e \underset{e}{\wedge} (\underset{k}{\vee} (K_k \& M_{k,g,e}))$$

$$E_e \Longleftarrow G_g \underset{g}{\wedge} (\underset{k}{\vee} (K_k \& M_{k,g,e}))$$

(where $\underset{i}{\vee}$, $\underset{j}{\wedge}$ are binary operators).

The procedure becomes clear if one imagines that the list is divided into sublists, and that for any given channel only one of the sublists is to be searched. The saving of time which results from such a procedure is evident.

One can also increase the throughput by taking into account the frequency of the forms, in that the list is arranged in decreasing order of frequency.

If the matrices are internally wired these considerations are no longer valid, as the acceleration factor would be about 1000.

There are several basic features which distinguish numerical from non-numerical data processing. Numerical data processing is generally characterized by:

A relatively small amount of input
" " " " " output
Extensive internal computation

Most conventional computers are designed for such tasks. On the other hand non-numerical work involves:

A large amount of input
" " " " output
A relatively small amount of actual computation
The use of a large storage capacity
No floating point computation

Forschungsgruppe LIMAS uses the computing center at the Institut für Instrumentelle Mathematik at Bonn University. The main programming language is FORTRAN II. Although FORTRAN was developed for numerical purposes it represents a relatively useful compromise between the various programming languages. The flexible subprogram system and the possibility of calling machine language programs from FORTRAN programs serve to facilitate the programming of non-numerical problems.

In order to simplify and expedite the programs, a computer with the following features would be desirable:

Byte-structure storage
Magnetic tape readers which operate in both directions
A large disk storage area
A high-speed printer with upper and lower case and special characters
A sophisticated wired addressing procedure

The last desideratum is justified in that non-numerical data processing is highly dependent on address manipulation. ~~For this~~ same reason, relative addressing, indirect recursive addressing and multiple addressing should be available. It should also be possible to annex extra hardware such as Dr. Hoppe's matrix. Wired search and sorting programs would also be practical.

With this sketch of some technical aspects of data processing I shall close my remarks. Thank you.

5

THE FULCRUM SYNTACTIC ANALYZER FOR RUSSIAN

Paul L. Garvin

The Bunker-Ramo Corporation
Canoga Park, California

1. The Fulcrum syntactic analyzer is part of a bipartite system for accomplishing automatic translation from Russian into English.

A bipartite system is characterized by the fact that it consists of two major portions: a machine dictionary and a translation algorithm. It differs from a tripartite system which consists of three major portions: a machine dictionary, a parsing algorithm, and a separate table of grammatical and semantic rules. In the author's opinion, the alleged advantages of a tripartite algorithm turn out to be illusory, whenever in practice the use of a realistically extensive grammar, and not just a few basic rules, is attempted. The grammar table then becomes so complex that it can no longer be handled by a simple algorithm. Then either the algorithm has to be adapted to the table, which reduces its generality, or a secondary algorithm has to be written which will show the processor the way around the table. The algorithm of a bipartite system is essentially a linguistic pattern recognition algorithm which, instead of matching portions of sentences against rules stored in a table, directs searches at the different portions of the sentence in order to identify its grammatical pattern. Thus, the essential characteristic of the algorithm is the sequencing of searches and, in each search subroutine, only as much grammatical information is used as is appropriate to the particular search. The rules of the grammar are in fact applied by the algorithm in a definite order, and a given rule is not even called unless the previous searches have led to a point where its application becomes necessary. This means that the highly complex system of rules that makes up the real grammar of a language is distributed over a correspondingly complex algorithm which applies the rules in terms of the ordering that the structure of the language requires.

The Fulcrum approach is based on two fundamental principles: the concept of the fulcrum and the pass method.

2. The concept of the fulcrum implies the use of key elements within the sentence (fulcra) as starting points for the searches performed by the algorithm. This means that the algorithm, in searching through a sentence, does not simply progress from word to word, but in fact "skips" from fulcrum to fulcrum. It performs a little search sequence each time it has reached a fulcrum, and goes on to the next fulcrum when this particular search is completed.

3. The pass method means that not one but several passes are made at every sentence, each pass designed to identify a particular set of grammatical conditions pertinent to the recognition process. Consequently, each pass has its own set of fulcra and its own search sequences. The pass method reflects the orderly progression in which the determination of the structure of the sentence is made: first, the sentence components are identified individually, then the relations between components are established, and finally the structure of the sentence as a whole is established. To each of these intermediate parsing objectives there corresponds, roughly, a pass or series of passes in the algorithm. The correspondence is not exact, because there are many ambiguities and irregularities interfering with the recognition process, and the design of the Fulcrum algorithm reflects these added complexities.

4. The Fulcrum syntactic analyzer has been implemented on two levels: (a) an earlier, less sophisticated version called Fulcrum I, which has been fully coded and is now capable of producing experimental translations; (b) a recent more sophisticated version, called Fulcrum II, for which detailed plans have been drawn up and which, when implemented on a general-purpose computer, will be capable of producing translations for practical use. It is expected that the complete implementation of the Fulcrum II will take four to six years.

5. The major improvements in the Fulcrum II over Fulcrum I are the following:

   a. The order of passes and individual search operations deviates further from the order of descriptive levels than was the case in Fulcrum I. This is due to the fact that the new search patterns are based primarily on the order in which the grammatical information becomes available to the program rather than the order in which a linguist would prefer to present his description.

   b. The Fulcrum II uses an iterative principle. The same set of search operations is used repeatedly to establish the internal structure of a variety of syntactic units. Thus, inserted structures (such as parenthetic expressions) are treated by the same search operation as entire sentences; classes of different types, such as relative clauses or independent clauses, are likewise treated by essentially the same search sequence. A control cycle insures that the different search sequences are called in the right order so that these different units are identified and related to each other appropriately.

c. The heuristic principle of trial and evaluation is applied throughout the Fulcrum II algorithm. In all parts in the program where decisions are not unequivocal, a capability will exist for labeling decisions as provisional trials, so that they can be revised later in the program by evaluation routines based on information available to the program subsequently.

6. The research principle of the Fulcrum approach is based on the interplay of theory and experimentation: by designing an experimental program based on an initial theoretical conception, this conception can be refined by computer experiments made possible by the availability of an experimental translation program at all times.

7. The theoretical basis for the Fulcrum approach is the Definitional Model of Language proposed by the author, the methodological foundations are those of the inductive method developed by the author.

OUTLINE OF FULCRUM-II SYNTACTIC ANALYZER

O. Dictionary lookup and stem-affixing

I. Pre-scan: paragraph numbering; determination of typographical sentence category

II. Pre-processing

1. Determine sentence boundaries; determine whether sentence is interrogative, or exclamatory. Otherwise leave unlabeled.

2. Assign grammatical function to symbols, non-Cyrillic matter, abbreviations (coded in dictionary according to whether or not they can be sentence-final), etc. [flag when necessary] .

3. Combine grammatical information on hyphenated words found separately in the dictionary.

4. Assign grammatical functions to missing words, initials and titles before last names (to form appositive nominal block with last name); include dictionary words ending in hyphen [flag when necessary].

5. Preliminary homograph resolution (include чем and как, but not commas or ambiguous conjunctions or dashes, nor potential interrogative pronouns, interrogative modifiers and interrogative adverbs) [flag when necessary] .

III. Pre-syntax

1. Set unambiguous clause boundaries using the following signals: subordinating conjunctions; relative and interrogative pronouns, modifiers, and adverbs; то after если and так как (check out comma), colons [flag when necessary]; if clause boundaries are set, label clauses as dependent or independent or relative according to established clause boundary and check whether both boundaries are ready to be set; otherwise leave unlabeled. If no clause boundaries are set, treat sentence as single unlabeled clause.

2. Label чем after comparatives as potential introducer of independent structure [flag].

IV. First processing cycle: *identify* inserted structures and parenthetic structures (including quotes), *label* them (from dictionary) as to function if possible (potential appositions, apposition introducers, complements, separate inserted sentences); this cycle includes a table lookup and, if the postential inserted structure is not found in the table, calls the main and final processing subroutines.

V. Second processing cycle: *process* dependent clauses (including relative clauses and indirect questions) in sequence, going right to left by calling the main processing subroutine.

VI. Third processing cycle: *process* independent and unlabeled clauses in sequence, going right to left (or *process* entire sentence as undivided whole) by calling the main and final processing subroutines.

MAIN PROCESSING SUBROUTINE, MPS
(detailed syntactic identification)

MPS-A Preliminary sequence: coordinate adverbs (semantic check); attach же and не to adverb, modifier, or pronoun; assign не to nominals, as applicable; package adverbs modifying other adverbs with their heads, creating adverbial blocks.

MPS-B Nominal blocking sequence

First subsequence: From right to left perform each step (1-3) as a separate pass at the clause or sentence.

1. Assign adverbs to modifiers or nominals as appropriate (including assignment of вот to demonstratives), label unattached adverbs.
2. Coordinate modifiers (semantic check), package modifiers modifying other modifiers (including самый) with their heads [flag] .
3. Assign modifier packages to nominals: label unassigned modifiers; label previously unlabeled unmodified nominals and personal, relative, and interrogative pronouns for further processing as nominal blocks.

Second subsequence: From right to left perform the following steps (4-10) in sequence, for each individual unassigned adverb, unassigned modifier, or nominal block, in a single pass.

4. Resolve genitive ambiguity when resolvable by preceding nominal, but not otherwise; [flag].
5. For nominal blocks only; attach prepositional structures and coordinate prepositional structures and appropriate adverbs if attachable to same head (include как...так и in coordination); attach governed structures including governed clauses; attach genitives of subject and object, attach other other genitive complements [set flag]; attach appositions; attach appropriate unassigned modifiers (including все, их; check commas) and governing modifier packages set flag when necessary (wipe out "missing mandatory head" label); coordinate governing modifier packages when applicable; attach complements and coordinate (include как...так и in coord.).
6. Coordinate nominal blocks and unassigned modifiers (wipe out "unassigned" labels for modifiers included

in the coordination), using semantic test; include appropriate unassigned adverbs in coordination (remove "unassigned" label); include constructions "nominal + c + nominal" and как...так и in coordination [set flag].

7. Label nominal block or unassigned modifier as prepositional structure if thus found; label nominal block or unassigned modifier as equational structure (e.g., как) if thus found; assign unassigned adverb or не to prepositional structure when applicable (wipe out "unassigned" label); assign apposition or appositional modifier to prepositional structure.

8. Assign relative clause (using existing clause boundaries as previously set) to antecedent nominal block; attach additional coordinate relative clauses if applicable.

9. Assign governed structures and complements (including complementary adverbs) to unassigned and/or governing modifiers, coordinate governed structures and complements when assignable to modifiers (including так...как и in coordination); assign governing modifier package to head if head is found to the right, but not otherwise (wipe out "unassigned" label); indicate missing mandatory head.

10. Assign dependent structure to unassigned comparative modifier or adverb when the latter is followed by чем ; assign dependent genitive structure to unassigned comparative modifier or adverb; assign prepositional structure to unassigned adverb; indicate missing mandatory head.

MPS-C Predicate blocking sequence

As a general search instruction, do not cross commas, conjunctions, or dashes, except when coordination is called in a step.

First subsequence: From right to left, perform all steps (1-7) as a single pass at the clause or sentence.

1. Label infinitives and gerunds.
2. Assign adverbs to infinitive or gerund.
3. Assign не, ни and reflexive pronoun to infinitive or gerund; assign past passive predicative to

быть and use government and complementation codes of past passive predicative in the subsequent steps; coordinate past passive predicatives when applicable and use government and complementation codes of last past passive predicative in further processing.

4. Assign objects and complements to infinitive or gerund block, coordinate governed and complementary prepositional blocks and adverbs when applicable (note whether infinitive or gerund can govern a modifier and in that case wipe out "missing mandatory head" or "unassigned" label from governed modifier); assign previously packaged что-clauses and indirect questions to infinitive or gerund.

5. Coordinate infinitive or gerund blocks; coordinate past passive predicative blocks assignable to forms of быть when applicable.

6. Assign infinitive block to previously established nominal block when applicable.

7. Indicate missing mandatory clause members when applicable.

Second subsequence: From right to left, perform all steps (8-15) as a single pass.

8. Label predicatives; label есть as special type of predicative.

9. Assign adverbs to predicative.

10. Assign не, ли, бы and reflexive pronoun to predicative; connect finite form of быть with predicative.

11. Assign objects and complements to predicative block [flag when ambiguous], coordinate governed and complementary prepositional blocks and adverbs when assignable to same predicative blocks (note whether predicative can govern a modifier and in that case wipe out "missing mandatory head" or "unassigned" label); cancel government code of comparative predicative when in чем...тем clause; assign previously packaged что-clauses and indirect question clauses to predicative.

12. Assign infinitive block to predicative block to form extended predicative block.

13. Coordinate predicative blocks (semantic check).
14. Find and label subject, or genitive or dative of reference (semantic check), for predicative block (include infinitive block, or что-clause or indirect question as subject, when applicable; do not cross other predicatives in this search).
15. Label clause members and include them in potential clause but do not absorb clause boundary in potential clause; revise ad hoc clause packaging done at III; indicate missing mandatory clause members and check whether candidates for these are not included in previously completed coordination—if so, revise packaging.

MPS-D Clause blocking sequence

As a general search instruction do not cross commas, conjunctions, or dashes, except when coordination is called in a step.

1. Find unattached infinitive blocks and check for dative of reference or бы , attach the latter to infinitive block, and label resultant package as predicative package and potential clause, as appropriate.
2. Check for missing mandatory head to modifier and attach modifier to head or relabel modifier as nominal, as appropriate; assign unattached adverbs, prepositional blocks and other potential complements (including special instrumentals) to either nominal to the right or entire clause, as applicable, and coordinate when applicable (including так...как и in coordination); include clause introducers in clause and indicate clause boundary where possible and not previously indicated.
3. Label previously unlabeled clauses as follows—decide whether что, чем and как introduce indirect questions; label что-clauses that are not indirect questions as such; label чтобы-clauses and special relative clauses (those separated by semicolons ?) as potential dependent structures, but retain clause status for purposes of attaching dependent clauses to them later; label infinitive clauses introduced by если, чтобы, прежде чем, перед тем как, etc., as dependent clauses; label clauses containing conjunctions and predicatives as dependent clauses; label clauses containing dative of reference and infinitive as independent clauses; label clauses containing бы and infinitives as dependent clauses; label clauses containing interrogative pronouns, modifiers or adverbs as direct or indirect questions, as applicable; label remaining clauses as independent clauses

FINAL PROCESSING SUBROUTINE (FPS):
SENTENCE BLOCKING SEQUENCE

Perform each step separately from right to left through clause or sentence.

1. **Resolve dash ambiguity (repeat predicative blocking sequence at MPS-C8 if dash is resolved as predicative).**

2. **Check for independent clause; interpret unattached commas, colons, and conjunctions as clause boundaries if surrounded by potential clauses (include flagged commas in search) or if likely, right boundary of relative clause—if not, check for other possible coordinations—and establish status of coordinated clauses as dependent or independent; resolve status of hitherto unassigned и and то (and similar particles).**

4. **Assign unattached dependent clauses to nearest whole clause, attach чем-clauses to тем-clauses.**

5. **Check functional potential of remaining unassigned structures and assign if possible.**

6. **Label inserted structures as potential nominal blocks, appositions, apposition introducers, potential complements, separate clauses or inserted sentences, as applicable.**

7. **Label unattached independent clause as whole declarative sentence.**

6

ANALOGIE ET HOMOLOGIE DANS L'ACCENTUATION DE L'ANGLAIS

par L. GUIERRE
Paris, Sorbonne , Mai 1967

ANALOGIE ET HOMOLOGIE DANS L'ACCENTUATION DE L'ANGLAIS

1. Terminaisons Faibles et Terminaisons Fortes

Le système accentuel de l'anglais (accent de mot) peut être décrit essentiellement, semble-t-il, à partir de l'opposition entre :

Terminaisons Faibles et Terminaisons Fortes

On appellera "forte" une terminaison qui impose un schéma accentuel constant (qu'il existe ou non une forme dérivante et quel que soit le schéma accentuel de cette forme dérivante). Ainsi <-ic> est forte puisque l'on a toujours /-20/ (accent principal sur l'avant-dernière syllabe) :

crǐtic, elěctric, enǐgma → enigmǎtic, mǎgnet → magnětic ... (1).

On dira que les terminaisons fortes sont la manifestation d'une force appelée analogie. Cela signifie, par exemple que tous les mots en <-ion> présents et, dans certaines limites, à venir, seront accentués /-20/ par analogie avec les quelques 2.000 mots attestés dans le corpus et accentués selon ce schéma.

On appelera "faible" une terminaison qui, ajoutée à un dérivant, n'en modifie pas l'accentuation.

Ainsi <-er> , <-able> , <-ly> sont faibles puisque l'on a

cǒmfort cǒmforter, cǒmfortable, cǒmfortably;
jǔstify jǔstifiable, jǔstifiably ...

On dira qu'un mot en <-ly> sera nécessairement accentué, par homologie, comme l'adjectif dérivant (adǒrably, fǒrtunately, immědiately, concǐsely...).

A partir de cette belle et pure dichotomie, qui fait penser à un language artificiel, on pourrait espérer que l'analyste n'aura même pas à hésiter ni à choisir dans sa description; tous les mots viendront docilement se ranger dans l'une des deux catégories (2).

Or cette dichotomie, même si elle est valable dans un très grand nombre de cas, ne résoud pas d'emblée tous les problèmes. On trouvera des "poles", des cas centraux : terminaisons faibles comme <-ful> , <-ness>, <-ing> , <-ed> , terminaisons fortes comme <-ity> , ou <-ical> , mais on verra qu'il existe fort peu de classes intégralement analogiques ou intégralement homologiques. Le plus souvent une dominante statistique très nette se manifestera, souvent le partage sera très inégal.

Exemple 1 : Les mots en <-ical> forment une classe analogique. Ils sont accentués /-200/. On trouve cependant au moins une exception (sur 600 mots environ) : le mot <umbilical> /1020/ [-ˈlaikl] (variante) qui ne peut être rattaché qu'à son homologue, le nom <umbilicus> /1020/ [-ˈlaikəs] (variante)...

Exemple 2 : La règle de "lion" : <-iV^{n}(C^{n})> ⇒ /-2(0)0/. Cette règle (3) signifie que l'accent /2/ (accent principal) précède immédiatement la terminaison graphique :

< i + une ou plusieurs voyelles +, facultativement, une ou plusieurs consonnes>; l'élision d'une syllabe, symbolisée par (0) étant gouvernée par des règles comme celles de la palatalisation et celles de la réduction de [i] à [ǐ] ou [j].

exemples : *[ˈimpiəs] → [impjəs] ; *[ˈglɔːriəs] → [ˈglɔːrǐəs] *[ˈaeksiən] → [akʃn].

Cette règle analogique et son corollaire régissent en fait 4.000 mots environ. Les finales attestées sont au nombre de 40 environ, les plus productives étant : <-ia>, <-ial>, <-ian>, <-iant>, <-io>, <-ion>, <-ious>, <-ior>, <-iot>, <-ium>. Les rares mots "irréguliers" (moins de 1 %) sont surtout, outre quelques emprunts étrangers non-assimilés (mafǐa, latakǐa, ratafǐa ...), des dérivés homologiques de verbes en <-y> : espǐal, decrǐal, defǐant, relǐant ...

Si <reliant> s'accentue /020/ [riˈlaiənt] c'est qu'il est rattaché à <(to) rely> /02/ [riˈlai]. Si <brilliant>, par contre, s'accentue /20/ [ˈbriljənt] et subit la pression analogique des mots en <-ia>, <-ian>, ... c'est qu'on ne peut le rattacher à aucun verbe.

La règle de "lion" et son corollaire s'appliquent aussi au schéma <-uV^{n}(C^{n})> (mots en <-uent>, <-uěnce>, <-ual>...) exemples : , influence [ˈinfluəns] → [ˈinflwəns], mellǐ-fluent; constǐtuent, accěn gargǎtuan, ǎnnual ... Les quelques mots irréguliers sont, s dérivés : subdǔe → subdǔal; spǐrit → spǐritual.

2.1 Forme et Fonction

Supposons une terminaison définie formellement par sa graphie (ainsi <-ive> ou <-ly> ou <-some>), il arrive souvent qu'à une seule forme graphique correspondent plusieurs phonies et/ou plusieurs fonctions syntaxiques (cf. l'exemple classique en français : les poules du couvent couvent)

Dans notre corpus on trouve :

attractive (adj.)	[-iv]	arrive (v.)	[-aiv]
supply (adv.)	[-li]	supply (v.)	[-lai]
troublesome (adj.)	[-səm]	chromosome (n.)	[-soum]

On trouvera, de même, que <-ish> terminaison adjectivante est faible (classe homologique) alors que <-ish> terminaison verbale est forte et produit une classe analogique.

Il peut donc être indispensable, pour définir complètement une terminaison, d'en donner la graphie, la phonie et la fonction si l'on veut éviter d'introduire certaines ambiguités dans la description de l'accent de mot. La recherche automatique a tenu compte de ces interférences entre les trois systèmes (10).

2. 2. Décomposition en classes

Ici encore la synchronie se distingue nettement de la diachronie. Les terminaisons <-ious>, <-ion>, <-ity> forment synchroniquement des classes homogènes (4). Cependant certains auteurs, influencés sans doute par des considérations historiques, proposent de décrire chacune d'entre elles en la subdivisant. Par exemple <-ious> est subdivisé en <-arious>, <-ocious>, <-itious>, etc... ce qui est inutile du point de vue qui nous occupe. Par contre, quand une classe comme <-ive> ne semble pas régie par un principe dominant (analogie ou homologie) on tentera, à juste titre, de la décomposer. On ne pourra analyser <-ive> synchroniquement qu'en distinguant <-ative>, <-itive>, <-utive>, etc... (voir plus bas parag. 4. 1).

2. 3. Conflits

Des conflits de nature diverse peuvent surgir :

a) Au sein même des modèles de dérivation, dans le cas où un dérivé peut être, à première vue, rattaché à deux dérivants d'accentuation différentes.

exemple : D. Jones donne pour <certification> soit /10020/ soit /01020/ (c'est le seul mot de ce type présentant cette opposition), mais il précise :

/10020/, act of certifying; /01020/, providing with a certificate. (variante).

Nous dirons : certify /200/ —> certification /10020/
certificate /0200/ —> certification /01020/

L'existence de variantes sera donc parfois (bien que l'occasion s'en présente rarement) justifiée par des considérations lexico-sémantiques. Cependant la notion de "related form" (5), de "proximité du dérivant" ou de "dérivant le plus proche" demande, on le voit, à être précisée.

b) quand une chaine de dérivations forme une "boucle".

Ainsi telephony fait partie de la classe analogique <-phony> /-200/. Si l'on choisit <-phony> comme point de départ on obtient par dérivation (6)

<-phony> /-200/	(-phonist (-phonize (-phonous (-....	/-200/ /-200/ /-200/	de même que	<-graphy> /-200/	(-graphist (-grapher (-....	/-200/ /-200/

ou encore

<-nomy> /-200/	(-nomist (-nomer (-....	/-200/ /-200/	et	<-scopy> /-200/	(-scopist (-scoper (-....	/-200/ /-200/

Par contre, de nombreux éléments monosyllabiques tels que <-phone> , <-graph> , <-nome> ... doivent être rangés parmi les terminaisons faibles (6) qui repoussent l'accent principal vers le morphème ou quasi-morphème initial.

exemples : tělephone, elěctrograph, dy̌namograph, mětronome ...

Dans le cas, heureusement ou plutôt nécessairement exceptionnel, où les mots en <-graph> , <-scope> , <-nome> , forment des verbes (to tělephone, to tělegraph) , on entrevoit immédiatement la latitude qu'aura la langue de créer, par dérivation, des termes en <-er >.

Mais alors aura-t-on telěphony → ? telěphoner /0200/
ou tělephone → ? tělephoner /2000/

ou encore telěgraphy → ? telěgrapher /0200/
ou tělegraph → ? tělegrapher /2000/

D. Jones répond : tělephoner et telěgrapher !!!

On peut schématiser cette situation ainsi :

(/200/ /-200/ /-200/
(-phony → -phoner, -phonist...

(-phone → -phoner ...
(/-0/ /-00/

Le conflit est latent (7) dans la mesure où deux mécanismes généraux de la langue produisent des effets contradictoires. Cette situation ne peut que freiner le développement de certains schémas de dérivation.

En fait, les verbes du type <telephone> sont peu nombreux et certains sont peu usités. Les dérivés en <-er> entrent d'ailleurs en concurrence avec les dérivés en <-ist> (c. a. d. ici, en <-phonist> , <-graphist> , <-nomist> ...) qui, eux, sont accentuellement déterminés. L'analyste sera donc amené à postuler que certaines dérivations doivent être orientées ou ordonnées, et à constater que d'autres sont préférentielles et d'autres encore condamnées d'avance comme non-viables.

c) quand la règle de dérivation aboutit à des structures accentuelles inacceptables.

Ainsi on a trouvé (8) que la séquence accentuelle /-12-/ n'était pratiquement jamais attestée dans le système de D. Jones. Une des rarissimes exceptions (ou bien est-ce une faute d'impression ?) est la variante /012/ de exchangee (également accentué /102/). En qualité de dérivé du verbe <exchange> /02/, on conçoit que <exchangee> conserve, sous forme d'accent secondaire, l'ancien accent principal du verbe. La finale forte <-ee> produisant l'accentuation /-2/, le résultat serait /012/ qui est, en principe absolument rejeté par le système. En fait le conflit devra être résolu c'est-à-dire qu'une règle devra l'emporter sur l'autre.

d) quand apparaît un danger de confusion formelle entre différentes terminaisons :

Ainsi pour la classe <-osis> l'analogie joue de façon rigoureuse (neurǒsis, psychǒsis, tubérculǒsis ... ont tous /-20/). Par ailleurs la classe de <-morphosis> , analogique elle aussi, produit /-200/. Cela semble indiquer que (de même que pour <-cracy> qui se sépare de l'ensemble {<-acy>} et s'en distingue comme une sous-classe homogène), la classe <-morphosis> doit être analysée indépendamment de celle des "autres mots en <-osis> ".

D. Jones donne deux accentuations pour <metamorphosis> :/10200/ et /10020/. La situation de <-morphosis> n'est en effet pas très simple. Des pressions latérales semblent s'exercer non seulement de la part des deux classes déjà mentionnées (<-morphosis> et "autres mots en <-osis>"), mais aussi de la part des terminaisons <-morphy> , <-morphous> , <-morphism> ... qui produisent toutes /-20/. La terminaison <-morphosis> est donc placée à un véritable carrefour d'influences inconciliables qui risquent fort de dérouter et l'étudiant étranger et l'indigène lui-même.

e) quand deux classes analogiques imposent des accentuations contradictoires.

Comment se comportent dans ce cas les mots qui appartiennent à la fois aux deux classes ?

On sait par exemple, que l'accentuation des mots en <-ic> est /-20/. On sait aussi que l'accent précède immédiatement la finale <-iV^n(C^n)>. Cette règle graphique, une fois transposée en termes phonétiques, reste valable, <i> devenant [i] , [ǐ] ou [j] . Comment réagissent les mots en [-iik] ?

On trouve <u>deux mots</u> de ce type : <ichthyic> , ['ikθiik] et <nucleic> , ['nju:kliik]. Ces deux mots se séparent de leurs analogues en <-ic> pour rejoindre les mots de la classe analogique <-iV^n(C^n)>. Nucleic et ichthyic ne peuvent évidemment pas satisfaire à la fois aux deux règles. La règle de "lion" domine la règle <-ic> ⇒ /-20/.

On verra, et l'on pouvait préssentir, que tous ces conflits sont nécessairement limités. Sinon la langue en tant qu'instrument de communication serait d'une complexité insupportable. Si, comme le montre l'expérience, ces conflits sont limités, c'est que la langue accepte un certain nombre de contraintes ou de règles du type par exemple : "le dérivé <evǒcative> ne peut être rattaché à la fois à <evǒke> et à <ěvocate> ; donc il ne devrait pas exister de couples de verbes du type <evǒke> et <ěvocate> (c'est en fait le seul cas attesté), et l'existence de <evǒke> bloquera le développement du type <ěvocate> (rare ?) et cela malgré l'existence attestée de <evocation>.

Par contre, certains conflits une fois résolus, c'est-à-dire dès qu'on aura établi la hiérarchie des règles rien ne pourra objectivement freiner le développement de certaines classes : appoint /02/ → appointee/102/ et donc exchange /02/ → exchangee /102/ , une fois rejetée la variante /012/ qui est contraire aux règles des structures accentuelles (cf. parag. 3. 2. 1.).

2. 4. <u>Absence de Dérivants</u>

Supposons que l'analyste, en se fondant sur les faits et les statistiques, décide de décrire une classe comme homologique : la majorité des mots de cette classe étant obtenue par dérivation comme c'est le cas pour <-ative> : <cǒncentrative> , <rǎdiative> , <remǔnerative> , <infǒrmative> , <exclǎmative>... L'expérience montre qu'il est rare que tous les dérivés possèdent un dérivant attesté (9). Il semble que lorsque le nombre de dérivants attestés en langue contemporaine "décroit" dans une classe donnée, cette classe tende à "devenir" analogique et, vice-versa, quand le nombre de dérivants attestés est élevé, la classe tende à "devenir" homologique. Il demeure que certains membres d'une classe nettement homologiques manquent de dérivants. L'analyste n'en est pas pour autant désarmé : d'autres procédés d'analyse s'offrent à lui. Il utilisera par exemple, non pas un processus de dérivation simple et linéaire (<apology> → <apologize>) mais le procédé des "réseaux" analogiques ou homologiques comportant au besoin des cases vides, c'est-à-dire, des formes "potentielles" (10).

Exemple : les mots en <-logy> , <-graphy> , <-scopy> , <-metry> , <-nomy> , <-gamy> , <-geny> , <-lysis> , <-thesis> , etc... sont tous accentués /-200/. Si on considère ces mots comme des formants déterminés analogiquement et, d'autre part, les terminaisons faibles : <-er> , <-ous> , <-al> , <-ent> , <-ence> , <-ize> , <-ist> , <-ism>... on peut établir un schéma générateur à deux dimensions qui permet de former des mots accentuellement déterminés en :

<-logist> , <-grapher> , <-gamous> , <-phonize> , <-scopal> ...

La langue contemporaine est loin d'utiliser pleinement toutes les possibilités de ce processus générateur, possibilités que nous considérerons comme faisant partie du stock disponible et accentuellement prédéterminé. De plus des "cases vides" peuvent apparaître en n'importe quel endroit de ce tableau y compris parmi les formants. On les remplira à l'aide de formes potentielles symbolisées (°) :

soliloquy → soliloquist , °soliloquent
°eloquy → °eloquist , eloquent
°antagony → antagonist , antagonism

Le même type de procédé permettrait d'expliquer la double relation :

discuss /02/ ↔ °discussive /-20/ ↔ discussion /-20/
°repercuss /102/ ↔ repercussive /-20/ ↔ repercussion /-20/

2. 5. Neutralisation

L'opposition entre analogie et homologie peut même se trouver neutralisée : certaines classes peuvent être décrites comme analogiques ou comme homologiques de façon tout aussi satisfaisante. C'est le cas de <-ical> qui peut être décrite soit comme une classe analogique (/-200/) soit comme une classe homologique de dérivés des membres de la classe analogique <-ic>. De même les verbes en <-ess> : <impress> , <regress> , <digress> ,

<progress>... /-2/ forment des adjectifs en <-essive> /-20/. Tous les dérivés en <-essive> partent de dérivants dont l'accentuation est uniforme. Dans ce cas le choix de l'analyse sera dicté par la possibilité de généraliser, d'étendre la règle au-delà de la sous-classe considérée.

3. 1. Les formants

La réduction des dérivés appartenant à des classes homologiques doit normalement produire des éléments de base que l'on appelera morphèmes, quasi-morphèmes, pseudo-morphèmes (11),(on néglige provisoirement la préfixation). Tous ces éléments seront appelés des "formants". L'analyste peut espérer que les formants seront régis :

a) par des règles analogiques qui sont des règles phonématiques séquentielles,

b) par des règles de structure accentuelle.

Ces dernières s'appliqueront à tous les mots dérivés ou non. Les règles phonématiques, au contraire, ne s'appliquent pas aux dérivés.

3.2.1. Règles de Structure Accentuelle - Les Quatre Contraintes

Ces règles sont de nature rythmique : elles concernent tout spécialement l'alternance des accents /0/, /1/, et /2/ dans le "squelette" accentuel des mots.

On a établi le tableau (11) des différents types de squelette et de leur population et l'on a été amené à poser quatre règles de base dont le degré d'exactitude dans le corpus étudié est de 99,8 % :

- 1°) Tout squelette comporte un accent /2/
- 2°) Aucun squelette ne peut comporter trois accents /1/ ou /2/ successifs en position quelconque.
- 3°) Aucun squelette ne peut avoir plus d'un accent /0/ à l'initiale : donc */00-/.
- 4°) L'accent /1/ est toujours suivi d'un accent /0/. Donc :
  a) */-12-/ B) */-11-/ c) */-1/

3. 2. 2. Conséquences : L'accent secondaire dans les mots non-composés

Les schémas possibles, c'est-à-dire autorisés par les 4 règles sont :

1 syllabe prétonique	/02-/
2 syllabes prétoniques	/102-/
au-delà de 2 syllabes prétoniques	/0102-/, /01002-/... /1002-/, /10002-/...

Comment est alors régie la position de l'accent /1/ (accent secondaire) au-delà de deux syllabes prétoniques ?

L'étude des mots de ce type triés automatiquement montre clairement que la place de l'accent /1/ est déterminée par l'accentuation propre du formant initial (morphème ou quasi-morphème). On sait que accent /2/ et voyelle neutre [ə] sont incompatibles. On a trouvé par ailleurs que jamais la voyelle neutralisée [ə] n'est attestée dans une syllabe frappée par l'accent /1/.

Exemples :

similar /200/	similitude /0200/	similarity /10200/
abolish /020/	[æ-] abolition /1020/	[æ-] abolitionist /10200/
hypothesis /0200/	hypothetical /10200/	hypothetically /102000/
[ə-] accept /02/	[ə-] acceptance /020/	[æ-] acceptation /1020/
[ə] academy /0200/	[æ-] academic /1020/	acadamician /01020/
experiment /0200/	experimental /01020/	experimentation /010020/
apologue /200/	apology /0200/	apologetically /0102000/
alcohol /200/	alcoholic /1020/	alcoholometer /100200/
microscope /200/	microscopy /0200/	microscopic /1020/
homonym /200/	homogeneous /1020/	homogeneity /100200/
heterodyne /2000/	heterogeneous /10020/	heterogeneity /100020/
epigram /200/	epigraphy /0200/	epigrammatic /10020/

3.2.3. Les Quasi-Morphèmes

On vient de voirque le Q. M. conserve un "écho" (accent /1/ au lieu d'accent /2/) de son accentuation d'origine quand il est suivi d'une terminaison forte (et après application des 4 contraintes). En outre, le Q. M. conserve son accent principal quand on lui adjoint une terminaison faible.

Exemples : bǐbliophile /2000/, phǒtograph /200/, cěphalopod/2000/ elěctrolyse /0200/, elěctrocute /0200/, psěudonym /200/, ǎtmosphere /200/... (voir aussi les exemples du parag. 3. 2. 2.).

Cela signifie naturellement que <electro-> par exemple, a une accentuation propre (/020-/) et qu'après application des 4 contraintes, et en l'absence d'une terminaison forte, on aura toujours /020-/.

Exemples : elěctrode, elěctrolyse, elěctron, elěctrophone, elěctroscope ...

En matière d'accentuation on a trouvé que les Q. M. se pliaient aux mêmes règles que les morphèmes autonomes (11) La règle de "lion" a été évoquée plus haut; une autre règle sera évoquée dans le paragraphe qui suit. (Parag. 3. 3. 2. 1).

3. 3. 1. Règles phonématiques

Rappelons que ces règles s'appliquent aux formants (morphèmes, quasi-morphèmes et pseudo-morphèmes) et pas aux dérivés. Certaines de ces règles ont été étudiées ailleurs (12) ce qui permettra d'être bref.

3. 3. 2. Règle des consonnes en contact

L'accentuation est /-20/ quand deux consonnes phonétiques en contact précèdent les terminaisons suivantes :

<-ive> , <-al> , <-um> , <-us> , (et <-ous>), <-a> , <-ent> , <-ence> , <-o> ...

Telle qu'elle est formulée ici, la règle comporte 4 % d'exceptions. On peut réduire ce pourcentage en éliminant [-kw-] et [-(s)tr] de la définition des "clusters". Il ne reste alors que quelques verbes préfixés : misgive, outlive, misdo, undo ... et deux ou trois dérivés :
sǔbstance -> sǔbstantive, sǐnister -> sǐnistrous

La règle a une portée plus générale qu'il ne parait ici et elle joue également pour les quasi-morphèmes (ex. : <elěctro->, <phary̌ngo-> mais hětero-, cěphalo-, etc...).

Les vérifications sont en cours, mais insistons sur le fait que cette règle ne s'applique pas aux classes homologiques, par exemple <-ingly> (dans rǎvishingly ...) ou <-bly> (dans mǐserably, disagrěeably ...) ou <-lessness> (dans cǎrelessness, pǎssionlessness...).

4. 1. Application - Terminaison <-ive> (résumé)

Cette terminaison est l'une des plus complexe et des moins régulières du système accentuel de l'anglais (population 850 mots environ).

4.3. NON-FORMANTS

Ce sont les mots de plus de deux syllabes. Ils constituent la grande masse des mots en ⟨-ive⟩ . La seule prononciation est [-iv] . Du point de vue de la formation, on peut souvent les rattacher aux mots des classes ⟨-ory⟩ et ⟨-ion⟩ .

A/ ⟨-ative⟩ (350 mots)

Trisyllabes :

Classe analogique /-200/

Modèle : laxative (ablative, narrative, locative, relative, talkative, causative ...).
Exception : creǎtive /020/ (dérivé de creǎte).

Notons que, quand le verbe dérivant est attesté, son accentuation n'est pas conservée (nǎrrative, rělative ...) d'où la décision de décrire cette classe comme analogique (1 exception).

Mots de plus de trois syllabes : Classe homologique

Modèle 1 : vǐndicate -> vǐndicative (gěnerative, pěnetrative ...)

Modèle 2 : consěrve -> consěrvative (imǎginative, quǎlitative, jǔstificative, intěrpretative ...).

Exceptions : indǐcative, altěrnative, interrǒgative, corrělative, demǒnstrative, signǐficative (cp. signǐficant).

Mots sans dérivants : /-200/ : prerǒgative, supěrlative, impěrative.

Cas douteux : evǒcative (dérivé de evǒke et non de ěvocate).

B/ ⟨-isive⟩ (150 mots)

Classe analogique /-20/

(la description par l'homologie serait presque aussi satisfaisante).

Modèle : exclǔsive (suspěnsive, dissǔasive, progrěssive ...)
Exceptions : pǔrposive (dérivé de pǔrpose seul formant nominal dans D. Jones; Walker donne aussi : rěcompensive (obsolete ?).

C/ <-C_1C_2ive> (200 mots)

(<-ive> précédé de deux consonnes graphiques différentes)

Classe analogique /-20/

Modèle : instǐnctive (assertive, detective, prescriptive ...)
Exceptions : ǎdjective, sǔbstantive (dérivé de sǔbstance).

Remarque : une description par homologie poserait le problème de la dérivation dans : incěntive, intellěctive, attěntive ... et celui des ajustements phonématiques peu uniformes dans :

prescrǐbe -> prescrǐptive; reděem -> reděmptive
presǔme -> presǔmptive; destrǒy -> destrǔctive ...

(la diachronie reprend ici ses droits).

D/ <-itive> et <-utive> — (60 mots)

Classe analogique /-200/

Modèle : gěnitive (additive, positive, punitive, intuitive, partitive, competitive, infinitive ...).
Exceptions : excǐtive [-aitiv] (venant de excǐte), rěsolutive (de rěsolute), sǔbstitutive (de sǔbstitute) sont des formations homologiques exceptionnelles car, d'autre part on a :

pretěritive (cp. prěterite), infǐnitive (cp. ǐnfinite) ...

De façon plus générale, cette classe se rattache à la classe phonétique des mots ayant [i] (ou [u]) en avant-dernière syllabe (<-ical> , <-iment> , <-itude> , <-imal> , <-itor> , <-ipal> , <-ulence> , <-ular> ...), classe phonétique qui se trouve en opposition avec celle des mots ayant [-ai] (ou [ɔi]) en avant-dernière syllabe (<-ival> , <-idal> , <-oidal> ...). On ne traitera pas ce problème ici.

4. 4. Mots hors-système :

khedive [ki'diːv] , recitative [ˌresitə'tiːv] .

5. CONCLUSION

Tel semble être, dans le cas de <-ive> , le programme, sous forme réduite, des structures maitrisées inconsciemment par l'indigène et, plus ou moins consciemment par l'étudiant étranger.

Redisons que d'autres terminaisons sont bien moins complexes. La terminaison <-ive> a permis de mettre en relief le conflit si fréquent entre analogie et homologie dans l'accentuation du mot anglais.

L'analyse automatique qui a permis d'obtenir ces résultats a été subventionnée par le C. N. R. S. et réalisée à l'Institut Blaise Pascal (C. N. R. S.) Paris.

L. GUIERRE
Paris, Sorbonne, Mai 1967

NOTES

1. 16 mots irréguliers sur 1. 600 (catholic, heretic ...)

2. - On pense à un partage d'influence entre une description néo-bloolfieldienne et une analyse Chomskiste. En fait, on le verra, il sera nécessaire d'élaborer une synthèse qui pourrait être une réconciliation et dont les parrains pourraient être A. Martinet, A. A. Hill et I. I. Revzin.

3. - cf. L. Guierre "Eléments ..." op. cit.

4. - voir, ci-dessus, parag. I exemple 2, la formule <-$iV^n(C^n)$> qui inclut <-ious>, <-ion> etc...

5. - cf. Arnold op. cit.

6. - cf. L. Guierre "Accentuation ..." op. cit.

7. - Même si, contrairement à ce que note Jones, il semble préférable d'écrire <-phone> /-1/ au lieu de /-0/ et <-phoner> /-10/ au lieu de /-00/.

8. - cf. L. Guierre "Accent secondaire ..." op. cit.

9. - Le dérivant peut avoir disparu tandis que le dérivé survivait, problème diachronique qui est écarté a priori de la présente étude.

10. - cf. L. Guierre "Un codage ..." op. cit.

11. - cf. L. Guierre "Eléments ..." et "Accentuation ..." op. cit.

12. - cf. L. Guierre "Accent secondaire ..." op. cit.

13. - Le cas de -ment terminaison faible substantive a été étudié dans L. Guierre "Eléments ..." op. cit.

BIBLIOGRAPHIE SOMMAIRE

ARNOLD (G. F.)	Stress in English Words, Lingua, VI, 4 Jul. 1957
FRIEDERICH (W.)	English Pronunciation, Longmans, 1958
GIMSON (A. C.)	An Introduction to the Pronunciation of English, Arnold, 1962
GUIERRE (L.)	Eléments pour une Etude Linguistique de l'Accentuation en Anglais, Les Langues Modernes, Paris, Sept. 1966 (Didier).
"	Accentuation, Dérivation et Syllabation, Les Langues Modernes, Paris, Sept. 1966
"	Un Codage des Mots Anglais en Vue de l'Analyse Automatique ..., Etudes de Linguistique Appliquée; n°4 Didier, Paris, Oct. 1966.
"	Accent Secondaire et Segmentation en Anglais, Les Langues Modernes, Juin 1967
"	Drills in English Stress-Patterns (à paraître chez Longmans, London, en 1968).
"	L'Accentuation en Anglais Contemporain (en préparation).
HILL (A. A.)	Introduction to Linguistic Structures, Harcourt, New-York, 1958.
JONES (Daniel)	English Pronouncing Dictionary, 12th edition, Dent, 1963
KINGDON (Roger)	The Groundwork of English Stress, Longmans, 1958

MARCHAND (Hans)	The Categories and Types of English Word-Formation, Otto Harrassowitz, Wiesbaden, 1960
REVZIN (I. I.)	Models of Language, Methuen, 1966
ROULLIER (U. F.)	Contribution à l'Etude Rationnelle de la Prononciation Anglaise, Orléans, 1937
WALKER (J.)	The Rhyming Dictionary, Routledge, 1957
WIJK	Rules for the Pronunciation of the English Language, O. U. P., 1966

⁂

7

SYNTACTIC AND SEMANTIC PROBLEMS IN AUTOMATIC SENTENCE GENERATION,

Kenneth E. Harper

A program for automatic sentence generation (ASG) has been developed in the linguistic group at The RAND Corporation, as a device for studying selected problems in syntax and distributional semantics. The general purpose of the study is to test certain linguistic assumptions by experimentation: what is the result when a computer program "composes" sentences on the basis of these assumptions? The assumptions, and their implementation in the computer program, are subject to modification, as deficiencies are observed in the output sentences. This trial-and-error procedure is presently at an early stage of development; the present paper is a brief, non-technical description of the procedure, and a report on some of the initial problems encountered, together with tentative solutions. The ultimate goal of the research is the generation of "paragraphs" (meaningful strings of sentences) as a contribution towards automatic abstracting.

1. The Sentence Generation Routine

The ASG program operates with Russian language materials, for the simple reason that the kind of data on which the linguistic assumptions are based is available only for Russian. The data is derived from the corpus of Russian physics text processed at The RAND Corporation (References 1, 2). In effect, the program described here deals with a sub-set of the Russian language found in these physics texts. The program may be said to consist of three parts: a glossary of words, a grammar, and a program for synthesizing sentences.

(i) The glossary is merely a list of 550 Russian words on magnetic tape; each entry consists of a "word number" (the word-identification number in the RAND physics glossary), (Reference 3) and a representative Russian form (not necessarily the canonical form). The glossary comes into operation only at the end of the sentence generation routine. Generally, the words in the glossary are used in a single sense in physics texts.

(ii) The grammar employed is a simplified dependency grammar for Russian. Its essential feature is that it is word-specific: each word in the glossary is provided a list of words with which it may combine in a dependency or governor relationship. The basic principle is that syntactic cooccurrence (e.g., the pairing of a given noun with a given adjective) is allowed only if this pairing is attested in the previously processed physics text. (An exception is made in the pairing of members of Semantic Classes, as explained below.)

The following information, available for each word in the glossary, may be said to constitute the grammar:

1) Word number
2) Part-of-speech code. Six such codes are used: VT (transitive verb), VI (intransitive verb), N (noun), A (adjective), DV (adverb), and DS (sentence adverb). A word may bear only one such code.
3) Semantic Class code (SC). Twenty such classes are recognized; all are distributionally formed. (References 4, 5)
4) Set of governing probabilities
5) Set of dependent probabilities
6) Coordinate probability

7) List of word numbers of governors
8) List of word numbers of dependents
9) List of word numbers of coordinates

Table 1 illustrates the meaning of governing probabilities. Reading across the top line in the table, we see that each VT has a probability of P_1 of governing some noun as subject and a probability of 1 of governing some noun as object. The VT has probabilities of P_2 and P_3 of governing an adverb and a sentence adverb respectively, and a probability of 0 of governing anything else. The table shows also that each VI must govern a noun as subject, and governs a DV and DS with probabilities P_4 and P_5 respectively; each noun governs another noun or an adjective with probabilities P_6 and P_7; a DS has a probability of 1 of governing a noun; adjectives and adverbs never govern.

Table 2 shows the meaning of dependent probabilities; here, reading across, the various probabilities of being governed by other parts-of-speech are shown for nouns, adjectives, adverbs, and sentence adverbs; verbs do not have governors.

The sets of governing probabilities for a word are associated with independent situations. For example, a transitive verb governs a noun as subject, a noun as object, an adverb and a sentence adverb independently. Therefore, in sentence generation, the decision to select a dependent type will be made without regard to any dependent types already selected for that governor. This is not the case, however, when selecting a governor for a particular word. The possibilities here are dependent events. An adverb, for example, must be governed by either a transitive or intransitive verb. Thus, the set of dependent probabilities for each word will add to one.

Probabilities for coordination are assigned for relatively few words. Since coordinate conjunctions are not represented as a word class, the assignment of a Russian form for "and" and "or" will be generated by the program (rather than by the glossary) in the sentence output stage.

As previously mentioned, each word is accompanied by a list of the word numbers that may function as its governors and dependents. Generally, these dependency pairs have been attested in physics text. However, a Semantic Class may also function as a word's governor or dependent; when this happens, the program is free to choose randomly any member of the SC in building the dependency pair. Many such pairs will not have been attested in text; the purpose here is to test the adequacy of the Semantic Classes in word combination. A complete list of the SC's cannot be given here; examples are (a) nouns that name physical properties ("height," "weight"), (b) verbs and nouns referring to a quantitative change ("to increase," "change"), and (c) the names of physical particles ("atom," "proton"). The classes, and their members, are purely experimental and are subject to modification.

(iii) The program for ASG (written in MAP for the IBM 7044) is intricately bound up with the grammar, but for purposes of discussion we may consider it separately. Essentially, the program has three functions. First, it restructures the grammar, in order to access it with minimal search time. (The grammar and the program are maintained in core storage.) Secondly, the program generates sentences; beginning at some arbitrary point, it proceeds, pair-wise, up and down the tree

Possible Dependents

Governor	VT	VI	N		A	DV	DS
VT	0	0	S P_1	θ 1	0	P_2	P_3
VI	0	0	S 1	θ 0	0	P_4	P_5
N	0	0	P_6		P_7	0	0
A	0	0	0		0	0	0
DV	0	0	0		0	0	0
DS	0	0	1		0	0	0

Table 1

Governing Probabilities

Possible Governors

Dependent	VT	VI	N	A	DV	DS
VT	0	0	0	0	0	0
VI	0	0	0	0	0	0
N	S P_1	S P_5	P_8	0	0	P_9
	θ P_2	0 0				
A	0	0	1	0	0	0
DV	P_3	P_6	0	0	0	0
DS	P_4	P_7	0	0	0	0

Table 2

Dependent Probabilities

structure until a terminal point is reached. In the process, the algorithm provides that decisions of two kinds will be made at each node: (i) shall a dependent (or governor) be chosen for the word at this node, and (ii) if so, which word shall be chosen to complete the dependency pair? The grammar supplies the basic information necessary for making decisions; a pseudo-random number generator with uniform distribution is used in conjunction with the grammar when a choice exists. Finally, the program prints out the generated sentences in a fixed format. In their preliminary form, the sentences are merely strings of word numbers, together with associated data; glossary lookup then provides the transliterated Russian form associated with each word number. At present, the problem of morphology is bypassed: no attempt is made to supply the correct inflection of words in the sentence. The information necessary to carry out an inflection routine is available for nouns and adjectives (i.e., inflection for case and number); for verbs, person and number are specified, but tense is not. In the sentences discussed below, correct forms for nouns, verbs and adjectives have been supplied for reasons of clarity.

An example will perhaps serve to clarify the operation of the ASG program. In a greatly simplified way, the algorithm proceeds as follows.

(1) A starting point is chosen for the sentence. Possible starting points are transitive verb (VT), intransitive verb (VI), and noun (N). By random selection, VI is chosen for this sentence.

(2) Randomly, a particular VI is chosen from the list of VI's in the glossary. Here, we assume that word number 56410 is chosen.

(3) The possible dependent types of a VI are considered. (According to the grammar, verbs do not have governors, so that only dependents need be considered.)

(3.1) The grammar specifies that a VI must have a subject (the probability P is one for this pairing). The list of dependents serving as subject for word number 56410 is consulted, and an individual word is randomly chosen: word number 34550.

(3.2) A VI may have an adverb (DV) as dependent. For the verb in our sentence, the probability for this event is found to be .5. We assume that in this sentence the decision is made that a DV will be selected. The list of dependent DV's for word number 56410 is consulted, and an individual adverb is randomly selected: word number 14090.

(3.3) A VI may have a sentence adverb (DS) as a dependent. For our particular verb, the probability for this event is found to be .1. We assume that a decision in made not to select a DS dependent.

(4) Next, the dependents of the dependents of the verb are considered, but only with respect to their possible dependents (i.e., working down the dependency tree structure).

(4.1) The noun chosen as subject (word 34550, from 3.1 above) may have various dependents, with varying probabilities. Each of these is considered in turn. For sake of brevity, we assume that only one is chosen in our sample sentence: an adjective. A particular adjective is randomly selected from the list of dependents for the noun: word number 63610.

(5) The adverb selected in 3.2 above has no dependents, according to the grammar, nor does the adjective selected in 4.1. At this point, the downward search for dependents is terminated, and the sentence is considered complete.

In form, the sentence is at this point a string of word numbers on a tape: 56410, 34550, 14090, 63610. Attached to each word number is data about its function in the sentence, its governor, the order in which it was selected, and, for nouns, an indication of grammatical number. Glossary lookup is then performed on this tape, and the transliterated Russian forms are printed out: uveličennoe otnošenie zametna èta.

If fixed rules relating to word order and morphology are applied to to this string of forms, the sentence emerges as:

Èto otnosenie zametno uveličivaetsja.

This ratio increases noticeably.

The foregoing is a drastically abbreviated description of the main steps in the ASG program. Sentences are presently generated at the rate of three per second; the addition of programming rules to account for morphology and word order would increase this time by an estimated ten percent. The program may, then, be considered as a practical, operational tool for research.

2. Discussion of Generated Sentences

At the present stage of development, the ASG program produces isolated sentences of varying degrees of complexity and "correctness." Since words in the glossary are limited in usage to one sense, and since semantic controls are guaranteed at least over pairs of words, a large number of sentences are quite acceptable. (The development of a context into which such sentences can be placed is, naturally, a far more difficult programming task.) The following are examples of "reasonable" sentences.

(1) Fejnman vyčislil integraly, s cel'ju opredelenija massy.
Feynman calculated the integrals in order to determine the mass.

(2) Rešenie zadač predlagaetsja v nastojaščej stat'e.
A solution of the problems is proposed in the present article.

(3) Ob"em kristallov izbytočnogo serebra bystro umen'šaetsja.
The volume of the crystals of excess silver rapidly decreases.

(4) Vozmožnost' sil'nogo vzaimodejstvija tela vnešnego istočnika privlekaet interes.
The possibility of the strong interaction of the body of the internal source is interesting.

Sentence (4) illustrates the approximate limit in number of levels (six) for "reasonable" sentences in the present system. Additional levels can easily be generated, but only through the process of annexing genitive noun modifiers (i.e., English "of" phrases). Very few sentences have been generated with more than six levels, principally because of the drastically reduced probabilities of noun complementation at lower levels in the tree. Thus, most sentences are short. The chief obstacle to increasing sentence length (and complexity) is, however, the absence in the grammar of provisions for subordinate and coordinate clauses. Also inhibiting, from this point of view, is the absence of participles, prepositional phrases (beyond those used as adverbs), and pronouns. The price for adding any of these grammatical categories is increased complexity in the program. In an experimental situation, brevity and stylistic monotony can be tolerated.

Deficiencies in the generated sentences are of two main types: syntactic and semantic. Problems in both areas will be illustrated, although the line of demarcation is sometimes difficult to draw.

Syntactic problems are chiefly the result of inadequate complementation of nouns by adjectives or other nouns. It will be recalled that the grammar specifies for each noun the probability of its modification by an adjective or a genitive noun. These probabilities are assigned on the basis of the noun's behavior in text (Refernce 6). Since P is normally less than one for both kinds of modification, a given noun may frequently appear in a generated sentence without modification:

(5) Stepanov ustanovil teorii zadač.
Stepanov established theories of the problems.

(6) Formuly tipa ispol'zujutsja.
Formulae of the type are used.

(7) Razrjad issledovan pri rasčete.
The charge was studied in calculating.

(8) Proverka daet metod polučenija atoma.
Verification gives a method for obtaining the atom.

In (5), the noun phrase, "theories of the problems," appears to be ill-formed; the difficulty may not really be syntactic, since in an appropriate context the phrase may be nothing more than an instance of ellipsis. Nonetheless, specificity as to "what kind of problems" should be provided in the given sentence, or in preceding sentences. The unmodified use of "type" in sentence (6) is more difficult to justify on the basis of ellipsis; our tentative solution is to require either an adjective or noun modifier for words like "type," "kind," etc. (A modification in the program is necessitated here, since at present the selection of adjective and noun dependents is made independently.)

The nouns in (7) and (8) are strongly verbal, and appear to be deficient in complementation. This is particularly true of "rasčete" in (7), translated with the "-ing" form because of its verbal usage with the sentence adverb, "pri." It should be said that all sentence adverbs in our grammar have the property of conferring a verbal function upon noun dependents. (This is, of course, a very limited use of sentence modifiers.) In order to maintain consistency in the grammar, it is clear that the program should be modified: the probability that a given noun will govern a genitive noun should equal one when the noun is itself governed by a sentence adverb.

In general, it is clear that a word's combining potential (i.e., its combining probabilities) may be affected by the syntactic environment in which it is placed. A generation program that does not take into account this possibility will be woefully inadequate. The problem is: which syntactic environments affect which words, and under which conditions? One use of the ASG program is to generate problems of this kind, and to test provisional solutions.

A second kind of syntactic problem arises when the grammatical number of nouns is inappropriate. (At present, the selection of number is made by the random number generator, operating on data in the grammar about the relative frequency of singular and plural in physics text.) Thus, in sentence (7) above, the combination "pri rasčetax" ("in calculations") could have easily been generated, since the noun, "rasčet," is frequently used in the plural. The strongly verbal nature of the noun in this environment, as noted above, makes the use of the singular noun almost imperative. (Deverbative nouns are almost never used in the plural when indicating a process.) The program should therefore be modified to require that a noun dependent of a sentence adverb be singular.

Two other instances of incorrect number in nouns may be mentioned. Nouns of the general classification, "abstract collective," require that genitive noun dependents be plural (unless the latter noun is rarely, or never, plural). Since the grammar contains no such specification, the following ill-formed sentence was generated:

(9) Čislo ètogo sloja otsutstvuet.
A number of this layer is absent.

Other nouns (deverbatives), and their corresponding transitive verbs, appear to require that the noun complement be plural.

(10) Izučenie stolknovenij atoma opublikovano v predyduščej rabote.
A study of the collisions of an atom was published in a preceding paper.

The solution to the problem is to modify the grammar so that "collison," will require the (subjective) genitive noun dependent in the plural. (A variation of this principle is "multiple complementation": "the collision of an atom with (and) another atom." Grammatical rules to account for this phenomenon are beyond the scope of the present grammar.)

We conclude that there are no major syntactic problems in the sentences so far generated, chiefly because the grammar is relatively primitive.

Semantic problems are more difficult to isolate. Again, it is sometimes possible that the absence of appropriate context is the chief cause of odd-sounding sentences.

A trivial kind of error is caused by the inappropriate repetition of a word:

(11) Analogičnaja formula i formula zapisalis'.
An analogous formula and a formula were written.

It is easy to forbid the repetition of a word in a sentence; such a restriction, however, would be too severe (cf., for example, "We used formula 1 in deriving formula 2."). In general, it appears that recurring words tend to be used in different clauses of the sentence. Until the program is capable of producing sentences of more than one clause, the best strategy is probably to forbid word repetition.

In some sentences, the choice of adjective modifier appears logically inconsistent, or incompatible.

(12) Molekuly detal'no izučeny, s cel'ju raznogo izmerenija.
The molecules were studied in detail, for the purpose of a different measuring.

The issue here is not the lack of a complement for "measuring," but the use of "different" with the strongly verbal noun governor. In one sense, the difficulty may be syntactic: "different" would in Russian normally be used with a plural noun, whereas a singular noun is strongly indicated in the present context. The oddity of the construction is, thus, partly the result of conflicting "forces" in the adjective and the noun.

Some sentences show the loss of meaning that may easily result from the expansion of noun phrases by the addition of genitive noun modifiers.

(13) Uravnenie zakonov raspada èffekta približenij polučen.
The equation of the laws of the decay of the effect of approximation is obtained.

The problem here is apparently the use of "effect" in two different phrases: "the decay of the (e.g., photoelastic) effect" is the kind of phrase found in physics texts, as is "the effect of approximating (e.g., the energy)." The confusion in (13) may be explained by the fact that the former expression refers to a physical phenomenon, whereas the latter refers to an exercise of the mind; the two incompatible phrases are forced together by the fact that "effect" is common to both. The anomaly may also be explained, more simply, by the different lexical properties of "effect."

A similar problem may also arise in the use of two adjective modifiers in a noun phrase. Thus, in the construction, "adjective/noun of adjective/noun," varying degrees of fitness may be observed for the adjectives:

All Usual New Incorrect Experimental	results of	(the) present theoretical detailed future	investigation show

Should some of these combinations be forbidden? If so, on what basis? What is the effect of expanding the grammar so that conditional clauses ("if . . ., then . . .") can be generated? One possibility is that adjectives can be semantically classified, so that logically incompatible combinations can be avoided. It is difficult to estimate whether or not such a path of investigation is fruitful.

At the present stage of research, problems of semantic "interference" have occurred infrequently. The generation of certain problem constructions is strongly indicated. For example, the program can be modified so as to produce, at a given point, one hundred sentences in which the construction, "noun of noun of noun," appears; likewise, it can be required that the second of the three nouns be a specific word (e.g., "effect"). From an examination of the output, we hope to gain some insight into the question of semantic compatibility.

REFERENCES

1. Hays, D. G., Basic Principles and Technical Variations in Sentence-Structure Determination, The RAND Corporation, P-1984, May 1960.

2. Edmundson, H. P., K. E. Harper, D. G. Hays and B. J. Scott, "Manual for Postediting Russian Text," Mechanical Translation, Vol. 6, 1961.

3. Kozak, A. S. and C. H. Smith, Studies in Machine Translation--12: A Glossary of Russian Physics, The RAND Corporation, RM-2655, October 1960.

4. Harper, K. E., Measurement of Similarity Between Nouns, The RAND Corporation, RM-4532-PR, May 1965.

5. Sparck-Jones, K., "A Small Semantic Classification Experiment Using Cooccurrence Data," M. L. 196, Cambridge Language Research Unit, January 1967.

6. Harper, K.E., Some Combinatorial Properties of Russian Nouns, The RAND Corporation, RM-5077-PR, September 1966.

SUMMARY - Syntactic and Semantic Problems in Automatic Sentence Generation

A computer program for the automatic generation of sentences was written, based on a simplified dependency grammar for Russian and a vocabulary of 550 words. Each word is provided a list of allowable constituents; the structure of a sentence is conditioned entirely by the lexical items chosen at each node in the dependency tree. Some of the generated sentences exhibit certain syntactic deficiencies (nouns are inadequately complemented or are given the wrong number); other problems result from the "interference" of semantic fields beyond the context of the word-pair.

Transformational Decomposition : A Simple Description of an Algorithm for Transformational Analysis of English Sentences*

Danuṭa Hiż

Aravind K. Joshi

University of Pennsylvania

ABSTRACT

In this paper, we will present a rather simplified description of an algorithm for transformational analysis (decomposition) of English sentences. Our purpose here is not to discuss the transformational theory, the full details of the theoretical formulations of the algorithm, or of the grammar. Rather, we will present a set of examples of the decomposition and some discussion of them with the hope that it will give enough insight into the capability of the algorithm and indicate to some extent the power of transformational analysis.

---

*This work was carried out in the Transformations and Discourse Analysis Project, University of Pennsylvania, sponsored by the National Science Foundation.

1.0 Here, we will present a rather simplified description of an algorithm for transformational analysis (decomposition) of English sentences. Our purpose here is not to discuss the transformational theory, the full details of the theoretical formulations of the algorithm, or of the grammar*. Rather, we will present a set of examples of the decomposition and some discussion of them with the hope that it will give enough insight into the capability of the algorithm and indicate to some extent the power of transformational analysis.

1.1 Transformations are certain relations among sets of sentences and in particular, it is possible to relate a given sentence to a set of elementary sentences (kernel sentences) by means of transformations. The kernel sentence forms (for English) are defined as the string of class marks N t V followed by one of the kernel object strings: φ, N, NN, NPN, ND, PN, D, A (N: Noun; t: tense/aux; V: verb; P: preposition; D: adverb; A: adjective; φ : zero). Thus John bought a book; Mary will come etc. are kernel sentences. Each transformation is characterized by certain permutations, deletions or additions of specific class marks or constants. In the resultant of a transformation one may look for subsequences which remain invariant even when the resultant is subjected to further transformations. The basic features of the algorithm are

a) stating the various invariant sequences and

b) formulating 1) a grammar of such invariant sequences, 2) a corresponding recognition procedure, and 3) a systematic procedure for computing the kernel sentences as well as other kernel-like sentences and the corresponding transformational history.

It should be emphasized that it is not assumed and also not implied in the algorithm that any kind of prior analysis (either string analysis or constituent analysis) is required as a prerequisite for the present algorithm.

---

*Such a detailed description will appear later elsewhere.

1.2 Transformations are initially defined on kernel sentence forms. However, they work on certain other sentence forms which are not kernel sentence forms. Thus a transformation is completely defined by first defining it on a suitable kernel sentence form(s) and then extending the deomain of the transformation to other sentence forms. This extension which contains infinitely many sentence forms can be represented by first listing a finite number of sentence forms in the extension and all the remaining sentence forms in the extension are obtained by certain recursive rules (see the i-lists in 1.3).

1.3 A unary transformation transforms one sentence form into another sentence form and a binary transformation transforms a pair of sentence forms into another sentence form. Each unary transformation defined on a sentence form may be represented by a sequence of class marks constituting another sentence form. Most binary transformations can be defined as interruptions of certain unary transformation sequences at stated positions by certain other sequences of class marks. These interrupting sequences are not sentence forms but are deformations of sentence forms corresponding to the second sentence form of the binary transformation. For example, *John was detained by the old woman* decomposes into *woman detained John* and *woman t be old* with a passive transformation on the first kernel and a binary transformation on the sentence *John was detained by the woman* and the kernel sentence *woman t be old*.[1] The sentence form corresponding to the passive transformation, *N t be en V by N* is then interrupted by the sequence *AN* before the last symbol. *AN* is a deformation of the kernel sentence form *N t be A* which is the second sentence form of the binary transformation. The resulting sentence form is thus *N t be en V by A N*. In the resulting sentence form the shared symbol N appears only once. Such a symbol which two transformation sequences share (or on which they overlap) will be called a residue of one sequence with respect to another.[2] In addition to the transformation sequences which are sentence forms, and the interrupting sequences (deformed sentence forms) which correspond to most binary transformations, there is yet another type of interrupting sequences (again deformed sentence forms) which correspond to nominalizations. For example consider: the book was written by Brown and John's travel to Italy was described by Mary. In the second sentence, the kernel sentence John travelled to Italy is mapped onto the object of Mary described before the resultant undergoes the same passive which acted on Brown wrote the book giving the first sentence. N's nV P N (John's travel to Italy) is a nominalization which appears in many different transformations and carries in them the associated kernel into one of the positions which could be occupied by a noun. For each transformation sequence in each intersymbol position we list all interrupting sequences (including both the second and the third kind of sequences as discussed above). Of course, the interrupting sequences have their own interrupting sequences, etc. These intersymbol interrupting lists will be called i-lists.

---

[1] We ignore here the article *the* for simplicity.

2. A sketch of the algorithm

2.0 As stated in 1, in order to define the set of all transforms we need a set of sequences of class marks (or class mark-like symbols) which has 3 subsets.

1. Sequences each of which corresponds to a sentence form (e.g. the passive sequence N t be en V by N);
2. Sequences each of which represents a deformed kernel-form and is not a sentence form, but when inserted between specified neighboring symbols of a sequence of the first set, preserves the character of the sentence form (e.g. AN, en V N);
3. Sequences each of which represents a deformed kernel-form and is

---

[2] The concept of the residue can be extended to shared sequences as well as sequences which replace a given symbol in another sequence. The term carrier is used in this context. This device has been extensively used in this algorithm.

not a sentence form, but, when substituted for a symbol in a sequence (of set 1 or 2 or 3), preserves the character of that sequence (e.g. er V or Ω, n A of N).

There are also rules for inserting sequences from the second set into other sequences or into sequences of the third set, without changing the character of either.

All insertion or replacement rules are stated in the interruption lists appearing between every pair of adjacent symbols of each sequence.

Most of the sequences in the first set represent unary transformations of kernel forms. Many are extended (often by permitting the replacement of certain symbols with selected sequences from the third set) to include analogous unary transforms of kernel-like forms.

The second set of sequences, together with the rules of their insertion in the sequences for unary transformations, account for most of the binary transformations. Other binary transformations are represented by replacement in pairs of class marks in unary transformation sequences by members of the third set, most of which consist of nominalizations.

An arbitrarily long English sentence form can be seen as composed of a finite number of such sequences recursively embedded in one another.

2.1 Corresponding to the above three subsets of sequences and their mutual embedding rules, we recognize three sets of strings. Each string is a program for comparing one of the sequences with a portion of the analyzed sentence form of the data. The program is equipped to permit interruption by other such programs according to the i-lists of the sequence. Each string, when entirely matched by a segment of data, replaces that segment with the carrier of the string. The carrier is sometimes null. In strings from the second set it is usually the residue of the binary insert (e.g. the center symbol of a noun phrase: N of AN, of ing V N,etc.). In strings from the third set the carrier is a class-mark-like symbol which, by replacing a classmark in a form derived from a kernel form, extends it to one similarly derived from a kernel-like form. Let the carrier be $\tilde{N}$[nV] for a noun phrase built around an nV. The extended passive form: N[or $\tilde{N}$[nV]] t be en V by N represents the form of the sentence John's travel to Italy was described by Mary as soon as the carrier of the string replaces in the data the nominal segment John's travel to Italy. The carrier from all strings in the first set is S, a symbol of a well-formed sentence.

The program of each string, whose sequence is a deformed (or transformed) kernel or kernel-like form, reconstructs that form for decomposition and attaches to it a label descriptive of the deformation (or transformation). The result of a decomposition is a set of kernel or kernel-like sentences with labels. Some of the kernel sentences are incomplete and have blanks in them because a transformation may delete elements. Some kernel-like sentences may contain, instead of a word, a class-mark-like symbol (e.g. $\tilde{N}$) with a reference to a previous component of the decomposition. If that previous component is a kernel sentence (with or without blanks), then the label (describing the deformation) with the kernel-like form (containing the reference) with its label, together constitute a description of the transformation undergone by the component kernel sentence. If the previous component itself is a kernel-like sentence with a reference in turn to another component, both kernel-like sentences and all three labels constitute the description of the transformation undergone by the component kernel sentence ultimately referred to, etc.

If the symbol x appears, instead of a word, in a kernel or kernel-like sentence, it replaces a regular noun there. It is introduced in the sentence as a carrier from a nominalization such as a teacher of Latin, the driving instructor, etc. The same x must appear in two or more sentences of the decomposition (one where the nominal stands for a noun, and one in the sentence of which the nominalization is a deformation, e.g. x - teach Latin). Which x's require identical substitutions is discoverable, because each x has in a sharp bracket (< >) the names of every previous line in which the same x appeared. Often no actual substitution is possible and the x serves only to identify, with each other, two or more blanks in different components. The substitution of the noun replacing N for x in lines a, b, . . . d is implied when one kernel-like component has the form N t be x <a, b . . . d>.

2.2 The three sets of strings (programs) constitute the major portion of grammatical material in the algorithm. Another body of such material is the dictionary.

The dictionary associates to each English word a symbol representing the word's grammatical class, together with markers of certain additional characteristics the word may reveal by restricting its environment in the sentence. Some words may occur in more than one role and have therefore several equivalents in the dictionary. (e.g. the word labor should be given four different class marks: present tense V, V(untensed verb), nV (nominalization designating the activity of laboring), er V (nominalization designating the actor(s), possibly laborers in aggregate)).

The dictionary for Transformational Grammar must carry far more details than is needed for the String Analysis alone. Thus for example the transformational analysis must be able to discover in John's sleep not only a noun phrase, but also the incomplete kernel sentence John-sleep φ which underlies each transformation containing such a noun phrase. Hence the class marks: nV (sleep), ing V (sharing) nA (bravery), erV (teacher), eeV (employee), nN (brotherhood), aV (helpful) and several others.

A V-entry in the String Analysis dictionary contains information about the kind of objects required by the verb V. An nV may require objects different from its V and this must be indicated (e.g. they attacked the enemy vs. they made an attack on the enemy).

Noun phrases like nV, ingV, etc. can occur in place of a sentence object or a subject of a sentence but only when it is organized around a verb requiring such subjects or objects, and such verbs are marked accordingly in the dictionary.

The subject and object restrictions for a verb or a verb-related word are recorded in pairs, because they are not mutually independent. (σ is the label for a subject (Σ) requirement; $\underline{\omega}$ for an object (Ω) requirement of a tensed or untensed verb and some ingV occurrences; $\underline{\omega_{nV}}$ labels an object requirement of nV-nominalization, $\underline{\omega_{ingV}}$ those of an ing V-nominalization, etc. When needed, $\underline{\omega}_1$ is distinguished from $\underline{\omega}_2$ (which usually is the same as the corresponding $\underline{\omega}$) to mark the form assumed by the object when it precedes the verb or verb related word (compare for instance house construction with construction of house where $\underline{\omega_{nV1}}$ (the same as ω) is N, while $\underline{\omega_{nV2}}$ (the same as $\underline{\omega_{nV}}$) is P [of] N).)

The analysis is preceded by a replacement of the words in the sentence by corresponding entries in the dictionary.

2.3 The process of analyzing a sentence begins in postulating (in turn) all those strings in the grammar which may occur at the beginning of a sentence (and whose initial symbol is the same as the first symbol in the data). (See i_1 of #30). Each verified postulate forces other postulates as its consequences, until the terminal period of the sentence is found which is consistent with a hypothesis. It is quite likely that an analysis will produce more than one correct reading of a sentence, because structural ambiguity is even more frequent in transformational grammar than it is in the mere string analysis.

3. Examples of decomposition

Four examples of decomposition obtained by the algorithm follow. These examples are intended to exhibit the power of the algorithm.

It is possible, without changing the algorithm, to increase the power and depth of the analysis by incorporating more details about transformations as they become available by adding either new transformation sequences or adding new classes and new co-occurence restrictions in the dictionary or both.

Among the various issues which are now receiving further attention, some are as follows: a) a better characterization of nF-nouns and the underlying kernel sentences in terms of which the modifiers can be explained (e.g. school principal (example 3), French teacher; British queen, etc); b) the relation of classifier nouns to each other and their kernel positions with respect to their modifiers (e.g. organic chemistry, helpful trip, friendly letter, etc.); c) precise relation of constants (e.g. his, both in example 4) or classifier nouns with a definite article to other nouns or phrases for which they are a replacement.

Examples: The first column lists the kernel sentences or kernel-like sentences (or intermediate resultants). The second column gives the rest of the transformational history. Here the names as stated are partial in the sense that the corresponding strings do not always correspond to complete transformatial sequences as discussed previously.

1.

Text:	The	fact	that	John	is	a	stranger	makes
	T	N_W	that	N	pres.be[3]	T	N	present V[3][1]
	his	life	here	unbearable.				
	R's	nV	D	aV				

---

[1] 3 indicates here 3rd person.

Decomposition:

	Kernel or Kernel-like sentences	transformation (partial names)	carrier
1.	John pres.be stranger (a)	container[2] noun: N_w that S	N_w ⟨1⟩
2.	He - live here;	$\tilde{N}$-nominaliztion; Σ's $nV\Omega$	$\tilde{N}$ ⟨2⟩
3.	- cannot bear $\tilde{N}$ ⟨2⟩	adjectivization : aV	$\tilde{A}$ ⟨3⟩
4.	N_w ⟨1⟩ pres. make $\tilde{N}$ ⟨2⟩ $\tilde{A}$ ⟨3⟩	container: N V_w NA	S

2.

Text:	Our	algebra	teacher	was	requested	by	the	school
	R's	N	erV	past be [3]	enV	by	T	N
	principal	to	interview	a	woman	candidate	from	Swarthmore.
	nF	to	V	T	N	N	P	N

Decomposition:

Kernel or Kernel-like sentences	transformations (partial names)	carrier
first reading:		
1. x - teach algebra	x-nominalization: ΩerV	x ⟨1⟩
2. We- have x⟨1⟩	left modified noun: N's N	x ⟨1,2⟩
3. x - heads[3] school	x-nominalization: NnF	x ⟨3⟩
4. woman- V_{app} P_{app} candidate (a) (V_{app} = be; P_{app} = φ)	compound noun: N_1N_2	candidate⟨4⟩
5. candidate-be from Swarthmore	noun, right modified: N_1 P N_2	candidate ⟨4,5⟩
6. x ⟨1,2⟩ - interview candidate ⟨4,5⟩	passive of container: N t V_w N infinitive	S
7. x ⟨3⟩ past request x ⟨1,2⟩ ⟨6⟩		

---

[2] Roughly, container forms are sentence forms in which 1) there is a verb (V_w) requiring a sentential subject or a sentential object or both or 2) there is a noun (N_w) or adjective (A_w) requiring sentential complements.

[3] heads is a $V_{appropriate}$ for nF principal as found in dictionary.

Kernel or Kernel-like sentences	transformations (partial names)	carrier
second reading:		
1. x - teach us algebra	x-nominalization: Ω erV	x $\langle 1 \rangle$
2. x - head school	x-nominalization: NnF	x $\langle 2 \rangle$
3. woman - V_{app} P_{app} candidate (a)	compound noun: N_1N_2	candidate $\langle 3 \rangle$
4. candidate-be from Swarthmore	noun, right modified: N_1PN_2	candidate $\langle 3,4 \rangle$
5. x$\langle 1 \rangle$ - interview candidate $\langle 3,4 \rangle$	passive of container: NtV_wN infinitive	S
6. x$\langle 2 \rangle$ past request x $\langle 1 \rangle \langle 5 \rangle$		

3.

Text:	Accident	insurance	of	an	employee	by	his	employer
	N	nV	P	T	eeV	P	R's	erV

protects		both.
present .	V: [3]	Q

Decomposition:

Kernel or Kernel-like sentences	transformation (partial names)	carrier
first reading:		
1. - - employ x	x-nominalization: eeV	x $\langle 1 \rangle$
2. x - employ him	x-nominalization: erV	x $\langle 2 \rangle$
3. x $\langle 2 \rangle$- insure x $\langle 1 \rangle$(an) P_{app} accident	$\widetilde{N}$-nominalization: nV	$\widetilde{N}[nV+\Omega+\Sigma]\langle 3 \rangle$
4. $\widetilde{N}$ $\langle 3 \rangle$ present protect both	container: $\widetilde{N}$ t V_w N*	S
second reading:		
1. - - employ x	x-nominalization: eeV	x $\langle 1 \rangle$
2. x - employ -	x-nominalization: erV	x $\langle 2 \rangle$
3. he - have x $\langle 2 \rangle$	left modified noun: $N'sN_2$	x $\langle 2,3 \rangle$
4. x $\langle 2 \rangle$ - insure x $\langle 1 \rangle$(in) P_{app} accident	$\widetilde{N}$-nominalization: nV	$\widetilde{N}[nV+\Omega+\Sigma]\langle 3 \rangle$
5. N $\langle 4 \rangle$ present protect both	container: N t V_w N	S

Note: The analysis would reach even deeper if the words his and both were treated as reference words leading to a substitution, e.g. of x $\langle 1 \rangle$ for he, x $\langle 1 \rangle$ and x $\langle 2 \rangle$ for both.

4.

Text:	Crop	sharing	between	the	tenant	and	the	land	owner
	N	ing V	P	T	N	and	T	N	erV

---

* Q may replace N.

is	an	economic	arrangement	unsatisfactory	to
present be [3]	T	aN	nV	aV	P

organized	labor.
enV	erV

Decomposition:

Kernel or Kernel-like sentences	transformations (partial names)	carrier
1. x - own land	x-nominalization: erV	x <1>
2. tenant (the) and x <1>(the)-share crop	$\widetilde{N}$-nominalization: ingV	$\widetilde{N}$[ingV + $\Sigma+\Omega$] <2>
3. - - arrange -; P_{app} economy	$\widetilde{N}$-nominalization;nV	$\widetilde{N}$[nv] <3>
4. x - labor -	x-nominalization: erV	x < 4>
5. - - organize x <4>	left modified noun: enVN	x<4,5>
6. $\widetilde{N}$ <3> -not satisfy x <4,5>	$\widetilde{N}$ right modified: $\widetilde{N}$ aV	$\widetilde{N}$ <3,6>
7. $\widetilde{N}$<2> present be $\widetilde{N}$ <3,6> (an)	container: $\widetilde{N}$t be $\widetilde{N}$	S

4. An illustration of the procedure

Example 5 John is a good story teller

This example illustrates the process of analysis in some detail. Because of space limitations for this paper a rather simple structure had to be chosen for this purpose. A short dictionary of the words in the sentence has been prepared and also a small set of grammar strings in provided for this illustration. Both were greatly simplified so that rich grammatical material will not obscure the demonstration of the choice of hypotheses, their verification or rejection, the use of the carrier, changes of levels in analysis and the exploration of alternative readings.

The analysis always begins with the string #30 postulated. A decomposition ends, when the program associated with this string is finished. All possible sentence beginnings are included in i_1 of #30. After the end of #30 alternative decompositions are sought.

When a new string is postulated on the basis of an i-list of another string, the verification of the new string takes place in the next level of a push-down memory, so that the state of computation of the suspended string is not affected.

Whenever two or more alternative paths open up for the analysis, each must be pursued to a successful completion or until failure occurs. (The analysis must produce every possible decomposition of a structurally ambiguous sentence). In our analysis, different paths are pursued serially. Every time an inspection of i-lists allows more than one hypothesis, one is chosen, while a list of the remaining ones together with all relevant positions of the memory goes on top of another push-down storage. The contents of that storage is examined after the end of the chosen path. The analysis ends after all possible paths have been explored and this storage is empty. In the example of analysis given here, we mark bypassed open branches by asterisks on the left margin and their resumption by similar asterisks encircled.

Dictionary used in Example 5.

John - N [proper, human, singular]
is - present be [3[1]; σ: N[or x]; ω; N/A/PN/D.
σ: $\tilde{N}$[nV/ingV]; ω: A_w, N_w, $\tilde{N}$[nV/ingV] etc.]
a - T[a]
good - A [A-ly = well]
story - N
teller - erV [σ : human, count; $\omega = \omega_{erV1}$: N/$\tilde{N}$[nV]/φ; ω_{erV2};
PN[or x or $\tilde{N}$: nV]/φ]

Grammar Strings used in example 5.

Nominal strings (each gives a noun-like carrier):

1. T[the/a/an]^{i_1} N[or x; or $\tilde{N}$: nV/ingV/nA/nN]^{i_2}
i_1 2,3,4,5
i_2 -
name: -
kernel: -
carrier: N[the article] (as matched)

2. A ^{i} N[or x]
i_1 2,4,5,
i_2 -
name: left modified noun: AN
kernel: N - be A (N,A as matched from data)
carrier: N< address of kernel> (N " " " ")

---

[1] Designates third person.

3. A ^{i_1} $\tilde{N}$[nV/ingV; or x]^{i_2}

i_1 3,
i_2 -

name: left modified nominal: A $\widetilde{N}$ [or x]
addition to kernel of $\widetilde{N}$: ; A-ly (A, as matched)
carrier: $\widetilde{N}$ ($\widetilde{N}$ as matched from data)

4. erV ^{i_1}
i_1 -

name: x-nominalization
kernel: x-V-
carrier: x [subclasses required from subject of V] ⟨address of kernel⟩

5. N[object] ^{i_1} erV ^{i_2}
i_1 -
i_2 -

name: x-nominalization
kernel: x- V N (N,V as matched from data).
carrier: x [subclasses required from subject of V] ⟨address of kernel⟩

6. nV ^{i_1} Ω_{nv} ^{i_2}
i_1 -
i_2 -

name: $\tilde{N}$-nominalization: nV[+Ω]
kernel: $\tilde{-}$ - V Ω_{nV} (as matched from data)
carrier: $\tilde{N}$[nV] ⟨address of kernel⟩

Object Strings:

10 - N[or x]^{i_1}
i_1 -

name - object
contribution to kernel in carrier
carrier: Ω[N] (N as matched in data)

11 - N[or x] ^{i_1} P ^{i_2} N[or x] ^{i_3}
i_1 -
i_2 1,2,4,5
i_3 -

name - object
contribution to kernel in carrier
carrier: Ω[N P N] (N P N as matched in data)

Sentence Strings

i_1 i_2 i_3 i_4
20 - N t V Ω
i_1 -
i_2 -
i_3 - 10,11,1,2,3,4,5,6
i_4 -

name: identity of kernel form: N t V Ω
kernel: N t V Ω (as found in data)
carrier: S < address of kernel>

i_1 i_2 i_3 i_4
21 - $\widetilde{N}$ t be $\widetilde{N}$
i_1 -
i_2 -
i_3 - 1,3,6
i_4 -

name: containing "be" ; $\widetilde{N}$ is $\widetilde{N}$
kernel: $\widetilde{N}$ t be $\widetilde{N}$ (as matched from data)
carrier: S < address of kernel>

Monitor String

i_1 i_2 i_3
30 - . S .
i_1 1,2,3,4,5,6,20,21
i_2 -
i_3 -

Illustration of the process of analysis:

Data: . N[John] pres.V[3,be] T[a] A[good] N[story] erV[teller] .
#30: . S . (level 1)
. = .
S $\neq$ N
i_1 of 30 allows the following strings beginning with N to interupt 30 here: 5,20. Try 20, mark * for the branch opening with 5 on level 2.

* Data: N[John] pres V[3,be] T[a] A[good] N[story] er V[teller]
#20 : N tV Ω (level 2)
N=N[John]
t=present
V=V be accepts John as subject. For a human subject, the object cannot be $\#\overline{6}$(in this simplified grammar). The verb be

rejects object form of #11.
Ω ≠ T Among the remaining strings of i_3 of 20 (1,2,3,4,5,10) only 1 starts by T.

Data: T[a] A[good] N[story] erV[teller]. (level 3)
#1 : T N
T = T
N ≠ A
i_1 of 1 has 2,3 beginning with A. Try 2, mark *** for bypassed 3.

*** Data: A[good] N[story] er V[teller]. (level 4).
#2 : A N
A = A
~~**~~ N = N note: i_1 of 2 has string 5 beginning with N. Mark ** for the bypassed branch.
end 2. kernel 1: story-be good. Resume 1.

Data: N[story] ⟨1⟩ erV[teller]. (level 3)
continue #1
~~***~~ N = N note: i_1 of 1 has string 5 beginning with N. Mark open branch ***. end 1. Resume 20.

Data: N[story] < 1 > (a) er V[teller]. (level 2)
continue #20
Ω ≠ N
of the strings from i_3 allowed by object requirement of the verb be, 5 and 10 begin with N. Try 10, mark **** for bypassed 5.

~~****~~ Data: N[story] ⟨1⟩ (a) erV[teller]. (level 3)
#10: N
N = N
end 10. Resume 20.

Data: Ω [N[Story] ⟨1⟩ (a)] erV[teller]. (level 2)
continue #20.
Ω = Ω [N[story] ⟨1⟩ (a)]
End 20. Kernel 2: John pres. be story ⟨1⟩ (a). Resume 30.

Data: S ⟨2⟩ erV[teller]. (level 1)
continue #30
S = S
· ≠ er
There is no string in i_2 of 30 which begins with er. Resume the nearest open branch: #5 at level 3. Erase mark **** (kernel 2 is also erased)

(****) Data: N[story] ⟨1⟩ (a) erV[teller]. (level 3)
#5 : N erV
N = N
er = er
V = V story is a proper object for teller

End 5. Kernel 2: x - tell story <1>(a). Resume 20.

Data: x <2>. (level 2)
continue #20
$\Omega \neq x$
The only string beginning with x among those of i_3 allowed as object of be is 10

Data: x < 2>. (level 3)
#10: N[or x]
N = x
end of 10. Resume #20.

Data: Ω[x < 2>]
continue #20.
$\Omega = \Omega$ [x[human, ct., singular] < 2>]. However, the verb be with a count-noun subject requires from a noun object an article or an article-replacer. This lacking, the current branch fails, the branch marked *** is reopened with #5 on level 4. (Kernel 2 of the failing branch is erased.) Erase ***.

*** Data: N[story] erV[teller]. (level 4)
#5 : N erV
N = N
er = er
V = V story is appropriate object for teller.
End 5. Kernel 2: x - tell story < 1>. Resume 1.

Data: x < 2 >. (level 3)
continue #1
N = x
end 1. Resume 20.

Data: x < 2>(a) . (level 2)
continue 20
$\Omega \neq x$
only one string beginning by x can interrupt here; it is 10.

Data: x < 2>(a). (Level 3)
#10 : N[or x]
N = x
end 10. Resume 20

Data: Ω[x < 2>(a)] . (level 2)
continue #20
$\Omega = \Omega$
end 20. Kernel 3: John pres. be x < 2>(a). Resume 30.

Data: S . (level 1)
continue #30
S = S
. = .
End 30

Print output:	1. story - be good	(left modified noun)
	2. x - tell story < 1 >	(x-nominalization: ΩerV)
	3. John present be x < 2> (a)	(identity of extended NtVN)

Are there any branches open? Yes, ** at level 5. #5 will be tried.
Erase **.

(**) - Data: N[story] erV[teller]. (level 5)
#5 : N erV
N = N
er = er
V = V story is appropriate object of teller
end 5. Kernel 1a: x - tell story. Resume 2.

Data: x <1>. (level 4)
continue #2
N = x
End 2. Kernel 2a: x <1> - be good. Resume 1.

Data: x <1,2>. (level 3)
continue #1
N = x
end 1. Resume 20.

Data: x <1,2> (a). (level 2)
continue #20
Ω ≠ x
The only string allowed to interrupt here is 10.

Data: x <1,2> (a). (level 3)
#10 : N[or x]
N = x
End 10. Resume 20.

Data: Ω[x <1,2> (a)]. (level 2)
continue #20:
Ω = Ω
End 20. Kernel 3a: John pres. be x <1,2> (a). Resume 30.

Data: S <3>. (level 1)
continue 30
S = S
. = .
end 30.

print output: 1. x - tell story (x-nominalization: ΩerV)
2. x <1> - be good (left modifier noun)
3. John pres. be x <1,2> (a) (identity of extended NtVN).

Are there any branches open? Yes, *** at level 4.
(To abbreviate, we will just say that this branch will be very much like the last one, except that, due to the difference between strings 2 and 3, it will give the output:

1. x- tell story; well (x-nominalication: ΩerV; left modified nominal)
2. John pres. be x < 1> (a) (identity of extended NtVN)

The last open branch, marked * fails immediately.)

References

1. Joshi, A. K., Hiż, D., "String representation of transformations and a decomposition procedure", Part I and Part II, Transformations and Discourse Analysis Project Paper, University of Pennsylvania, Dec. 1965.

2. Joshi, A. K., "Transformational analysis by computer with some applications", Presented at the National Institute of Health Seminar on Computational Linguistics, Bethesda, Oct. 1966. (To be published).

9

Research Group for
Quantitative Linguistics
Fack
Stockholm 40
SWEDEN

KVAL PM 337
June 19, 1967

SLANT GRAMMAR CALCULUS

By

HANS KARLGREN

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

SLANT GRAMMAR CALCULUS

By

HANS KARLGREN

KVAL, Fack, Stockholm 40, Sweden

Summary

Bar-Hillel and Lambek have outlined a syntactical description where the syntactical category symbols are written as fractions and where the analysis of a given sentence is performed according to rules very similar to common arithmetical reduction of fraction expressions with factors that cancel out. Are there algorithms for applying such a calculus to normally complex natural language structures?

1. Aim

We seek a formal recognition procedure that will enable us to decide for any given sequence of elements from a given language whether or not the sequence is grammatical.

We consider only one method, named that of a categorial grammar (Bar-Hillel and Lambek).

We make the following assumptions:

(a) The knowledge we want to utilize for recognition can without residue be summarized in

> a list, giving for each word the grammatical categories the word belongs to; a set of combination rules for the category symbols.

(b) A sequence of elements is grammatical if there exists at least one word-for-word translation of it into grammatical category symbols which yields a symbol sequence that is permitted according to the set of combination rules. We say that a symbol sequence which agrees with the combination rules is a grammatical symbol sequence.

2. Shrinking Procedure

We assume that it is possible to verify the grammaticality of a symbol sequence by reduction of it to simpler and shorter sequences step by step. In each step one or more symbols in the sequences are replaced by one new symbol.

The string replaced by one other symbol will - to begin with without linguistic interpretation - be called a syntagm; the replacing symbol will be called the name of the syntagm.

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

By successive application of rewriting rules, the original sequence is shrunk to a no longer reducible residue, which may be just one symbol. If this residue is contained in a given list of permissible sentence patterns, the sentence is grammatical.

3. Slant Grammar Calculus

The kind of grammar under study we shall simply call slant grammar from its salient trait, the notation. It is characterized by the following properties:

a) The category symbols are all of the form

$$a \qquad \text{(atomic symbols)}$$

$$\text{or} \quad \left.\begin{matrix} x/y \\ y\backslash x \end{matrix}\right\} \qquad \text{(complex symbols)}$$

where x and y in their turn have the same form (atomic or complex) as the category symbols. We shall call a and x numerators and y a denominator in such cases.

b) Combinatorics is condensed to the following

α) a symbol sequence is a grammatical syntagm of type t if and only if it can be reduced to t by successive application of one of the following two cancellation rules for contracting two neighbouring symbols of the - original or the so far reduced - sequence into one symbol:

$$x/y\ y \to x$$
$$y\ y\backslash x \to x$$

where x and y are atomic or complex symbols.

β) a grammatical sentence is a syntagm of a type which belongs to a short list of possible types of patterns, say type s.

The categorial notation seems helpful in establishing a recognition calculus. Some programmable algorithms will be discussed in this paper.

It is easily seen that slant grammars of the type discussed are equivalent to context-free phrase structure grammars (as far, i.e., as any generative grammar can be "equivalent" to a recognition grammar).

The cancellation rules presuppose that if the symbols a/b and b are reduced to a, there must not stand anything between the syntagms a/b and b - i.e., there must be no hole in the syntagm a - although the symbols a/b and b may in the original sequence stand widely apart.

A slant grammar for one given language may be written in many different ways. Thus one may design the grammar so that the category symbols have at the most one denominator and even so that they have only left denominator or only right denominator (Marcus).

A natural way to design the grammar would be to let governed syntagms have simple symbols (y) and the governing ones complex symbols x/y, or inversely, so that the relation operator/operand would imply dependency relation.

However, the number of alternative symbols for each word will tend to increase if such a priori rules should apply to the whole set of symbols. Given the algorithm for recognition, one may ask how the categorial grammar should be designed so as to give the minimum number of operations, e.g., so as to yield on an average, the minimum number of possible word-for-word translations into grammatical symbols.

4. Reduction Procedure

To begin with, we shall investigate some procedures for analysis of a given sequence of category symbols. Then, cf. 8 below, we turn to the practically more important problem when not a sequence of symbols but a sequence of words is given, each word having several potential categories.

5. Substituting Complex Symbols

We make a preliminary simplification of the problem by replacing every complex denominator in the sequence by a new, arbitrary atomic symbol. Simultaneously, we make corresponding substitutions of numerators: if we replace b/c by $\underline{x}$ as a denominator, we also replace b/c by $\underline{x}$ at some other place, where b/c appears as a numerator.

Example:

$$b/(c/a)\quad c/a\quad b\backslash a/(b/c)\quad d\quad d\backslash(b/c) \Rightarrow b/y\quad y\quad a/x\quad d\quad d\backslash x.$$

Now, if b/c should happen to appear in numerator position more often in the given sequence than it does qua denominator, this replacement can be performed in more than one way. We then do perform it in more than one way, thus generating a number of alternative symbol strings to be processed. Through this artifice, we have sequences where all denominators are certain to be atoms, a fact which radically simplifies the analysis. Instead we have made the symbol selection procedure more difficult.

Since now all denominators are atoms and since $(b\backslash a)/c$ is equivalent to $b/(a/c)$ and to $b\backslash a/c$, the brackets are now redundant and can be omitted.

The symbols, then consist of a kernel atom, possibly neighboured at one side or both by a *slant* and another atom, in its turn possibly neighboured by slant plus atom, and so on, all slants to the left of the kernel being tilted to the left and those to the right tilted in the opposite direction:

$$a,\ a/c,\ b\backslash a/c,\ g\backslash f\backslash e\backslash d\backslash a/b/c/,\ \ldots$$

If one knows which element is the kernel,one does not even need the slants and we can proceed to simplify one step further:

$$\underline{a},\ \underline{a}c,\ b\underline{a}c,\ gfed\underline{a}bc,\ \ldots$$

where the underlined characters are numerator atoms and all others are denominator atoms.

(If the language has no rules for the relative order of syntagms, grammaticality is rapidly tested. Just check that to each denominator atom corresponds one numerator atom of the same name, leaving without a match just one numerator, which then denotes the type of the syntagm. In this case we are permitted to treat the atoms as numerators and denominators in the *arithmetical* sense. If we assign prime numbers to each atom and reduce in the standard arithmetical way, we end up with the numerical value of the type of the syntagm. This simple test may be worth considering as a first check, even though the structure of the language be far more complex.)

6. Wanted: Non-Intersecting Vaults

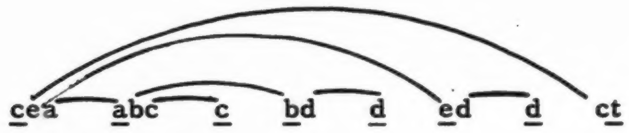
The two above-mentioned cancellation rules are, rewritten in the modified (simpler) notation, fused into one:

> If in a string of characters, an underlined character and a non-underlined character appear, separated only by a space character and denoting the same atom, erase all three characters.

Our task may also be formulated thus:

> Draw a connecting line from each denominator in the string to a numerator with the same name, in the correct direction - i.e., not over the numerator of the own symbol - like vaults over the strings, in such a way that no two such vaults intersect and that one numerator - say t - remains untouched by any vault nor roofed by one. If this succeeds, the string is a grammatical sequence of type t.

Example:



7. Identification of the Correct Numerator

If all numerator symbols are different the task is trivial. If several numerator atoms carry the same name, we have to decide which one is intended by a given denominator. How do we do this?

The problem is in this form a purely computational one - and no easy such problem.

8. Algorithm with Stacking

One algorithm can be summarized as follows:

We have a given string of words and want to ascertain whether its type is one of the set T = {s, t, ...}. For each word we have a given set of alternative category symbols.

We first consider the simpler problem of analyzing a given string of category symbols and see if it can be reduced to s.

We join the (not underlined symbol) s followed by space to the beginning of the given symbol string. When the reduction rules are applied the resultant string should vanish; we say it is reduced to unity.

The string now contains exactly as many, say n, numerator and denominator atoms. Every numerator should be paired with one other denominator; the whole problem is to decide which denominator.

Whenever we thus pair two atoms, the intervening string of atoms should vanish under reduction, be reducible to unity. The same holds for the rest of the original string, after the two paired atoms and everything between them has been removed:

$$s \quad \underline{a} \quad a\underline{b}cd \quad \underline{d} \quad \underline{e} \quad e\underline{c}fg \quad \underline{g} \quad \underline{f}$$

In general, then, when one wants to test whether a pairing is a good match or not, one is left with two simpler, isomorphic problems, those of reducing two strings to unity:

$$s \quad \underline{a} \quad a\underline{b}fg \quad \underline{g} \quad \underline{f}; \quad d \quad \underline{d} \quad \underline{e} \quad e$$

Normally we cannot decide beforehand which pairing to test; we must try alternatives. For each arbitrary choice of procedure, we store the alternatives in an OR-stack, For each pairing we place the bracketed-out string in an AND-stack, to be handled later when and if the string on our working table vanishes. If we have emptied the table and the AND-stack, the given string was reducible to s. If we have an irreducible residue on the table, we clear the table and try the next alternative if there is one in the OR-stack, deleting what has been stored in the AND-stack since the last arbitrary choice; if we cannot reduce to unity the string on the table and the OR-stack is empty, we must give the task up.

To minimize the number of pairings to test for each string we analyze, we draw up a binary matrix M of potential matches, with $m_{ij} = 1$, if the denominator No. i and the numerator No. j are the same character and are placed in proper order and reasonably wide apart in the string. (E.g., if $\underline{i}$ is a "non-recursive" type of syntagm, "reasonably" placed means that $\underline{j}$ is the *nearest* numerator written with the same character as $\underline{i}$ and placed on the proper side of $\underline{i}$.)

If any row or column in M is empty, the task is hopeless. Otherwise, we select the atom whose row or column contains the least number of 1's.

To analyze a string of words we assign to each word one symbol of the type $AB\underline{C}DE$, where C is the set of numerators in all the word's category symbols, B is the set of nominator atoms appearing immediately before the numerator in any one of the word's category symbols, etc. On the string of these new symbols the above procedure, mutatis mutandis, is applied. Thus, instead of the condition in (1) above that two atoms should be the "same" characters, it is required that one underlined atom A and a non-underlined atom B appearing separated only by space should fulfill the condition $A \cap B \neq 0$.

References

1. Y. Bar-Hillel, A quasi-arithmetical notation for syntactic description. *Language 29*, 47-58 (1953).
2. J. Lambek, On the calculus of syntactic types, in "Structure of Language and Its Mathematical Aspects". *Proc. 12th Symp. Appl. Math.*, American Mathematical Society, Providence, R.I., 1961, pp. 166-178.
3. S. Marcus, Algebraic Linguistics; Analytical Models, New York & London, 1967.

10

EXPERIMENTS WITH A POWERFUL PARSER

by

Martin KAY

The Rand Corporation
1700 Main Street
SANTA-MONICA - California 90406 - U.S.A.

EXPERIMENTS WITH A POWERFUL PARSER

This paper describes a sophisticated syntactic-analysis program for the IBM 7040/44 computer and discusses some of the problems which it brings to light. Basically the program is a nondeterministic device which applies unrestricted rewriting rules to a family of symbol strings and delivers as output all the strings that can be derived from members of the initial family by means of the rules provided. A subsidiary mechanism deals with the relation of dominance in the sense common in linguistics. This makes it possible for rules to refer to complete or partial syntactic structures, or P-markers, so that the program can be used at least to some extent for transformational analysis.

A program of this kind, which is intended for analysing natural languages, must be capable of operating on a family of strings as a single unit because of the grammatical ambiguity of words. Take, for example, the famous sentence "Time flies like an arrow." These five words are not, themselves, the primary data on which a parsing program can be expected to operate. Instead, each word is replaced by one or more symbols representing the grammatical categories to which it belongs. The assignments for this example might be somewhat as follows:

Word	Grammatical category
Time	Noun, verb, adjective
flies	Plural noun, 3rd person verb
like	Singular noun, preposition, verb
an	Indefinite article
arrow	Singular noun, adjective.

Taking one category symbol for each word, it is possible to form 30 different strings, preserving the order of the original sentence. These 30 strings constitute the family on which the program would operate if set to analyze this sentence.

The program is said to perform as a non-deterministic device because whenever two mutually incompatible rules are applicable to the same string neither is given any priority; both are applied, and the resulting strings developed independently. Given the string "A B C" and the rules

A B → X Y

B C → Z

the program will therefore produce two new strings:

X Y C

A Z

The program contains no mechanism for guarding against sequences of rules which do not terminate. If the grammar contains the following rules

A B → B A

B A → A B

and the string to be parsed contains either "A B" or "B A," then the program will continue substituting these sub-strings for one another until the space available for intermediate results is exhausted. This may not seem to present any particularly severe problem because a pair of rules such as these would never appear in any properly constructed grammar. But, as we shall shortly see, entirely plausible grammars can be constructed for which this problem does arise.

1. THE FORM OF RULES

In order to get a general idea of the capabilities of the program, it will be useful first to consider the notation used for presenting rules to it and the way this is interpreted by the machine. In what follows, we shall assume that the reader is familiar with the terminology and usual conventions of phrase-structure and transformational grammar. An example of the simplest kind of rewrite rule is

VPRSG = PRES SG VERB

The "equals" sign is used in place of the more familiar arrow to separate the left and right-hand sides of the rule. The symbols on which the rules operate are words consisting of between one and six alphabetic characters. The above rule will replace the symbol "VPRSG" by a string of three symbols "PRES SG VERB" whenever it occurs. The following rule will invert the order of the symbols "VERB" and "ING"

VERB ING = ING VERB

The simplest way to represent a context free phrase structure rule is as in the following example:

NP AUX VP = S

Notice that the normal order of the left and right-hand sides of the rule is reversed because the recognition process consists in rewriting strings as single symbols; the rules must therefore take the form of reductions rather than productions.

The program will accept phrase structure rules in the form we have shown, but, in applying them, it will not keep a record of the total sentence structure to which they contribute. In other words, it will cause a new string to be constructed, but will not relate this string in any way to the string which was rewritten. One way to cause this relationship to be preserved is to write the rule in the following form:

NP.1 AUX.2 VP.3 = S(1 2 3)

The number following the symbols on the left-hand side of the rule function very much like the numbers frequently associated with structural indices in transformational rules. When the left-hand side of the rule is found to match a particular sub-string, the number associated with a given symbol in the rule becomes a pointer to, or a temporary name for, that symbol. With this interpretation, the left-hand side of the above rule can be read somewhat as follows "Find an NP and call it 1; Find an AUX following this and call it 2; Find a VP following this and call it 3."

The numbers in parentheses after a symbol on the right-hand side of a rule are pointers to items identified by the left-hand side, and which the new symbol must dominate. In the example, the symbol "S" is to dominate all the symbols mentioned on the left-hand side.

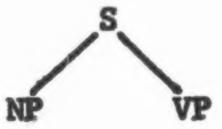
A pointer may refer to a single symbol, as we have shown, or to a string of symbols. The following rule is equivalent to the one just described:

NP.1 AUX.1 VP.1 = S(1)

Furthermore, the string to which a pointer refers need not be continuous. Consider the following example

NP.1 AUX VP.1 = S(1)

This will cause any string "NP AUX VP" to be rewritten as "S", but the "S" will dominate only "NP" and "VP." There will be no evidence of the intervening "AUX" in the final P-marker which will contain the following phrase:

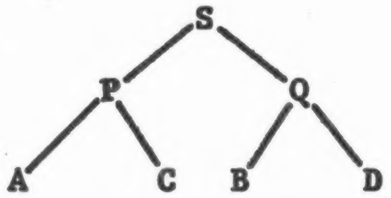


Consider now the following pairs of rules:

A.1 B.2 C.1 D.2 = P(1) Q(2)

P.1 Q.1 = S(1)

If these rules are applied to the string "A B C D" the following P-marker will be formed:



Notice that the first rule in the pair not only reorders the symbols in the P-marker but forms two phrases simultaneously.

A different way of using pointer numbers on the right-hand side can be illustrated by comparing the effects of the following two rules:

N.1 SG.1 V.2 SG.3 = NOUN(1) V(2) SG(3)

N.1 SG.1 V.2 SG.2 = NOUN(1) 2

What is required, we assume, is a context sensitive phrase structure rule which will rewrite "N SG" as "NOUN" in the environment before "V SG". The first rule achieves this effect but also introduces a new "V" dominating the old one, and a new "SG". The second rule does what it really wanted: It constructs phrase labeled "NOUN" as required, and leaves the symbols referred to by pointer number 2 unchanged.

The context sensitive rule just considered is presumably intended to insure that singular verbs have only singular subjects. A second rule in which "SG" is replaced by "PL" would be required for plural verbs. But, since agreements of this kind may well have to be specified in other parts of the grammar, the situation might better be described by the following three rules:

```
SG.1 = NUM(1)

PL.1 = NUM(1)

N.1 NUM.2 V.3 2 = NOUN(1 2) 3 2
```

The first two rules introduce a node labeled "NUM" into the structure above the singular and plural morphemes. The third rule checks for agreement and forms the subject noun phrase. Pointer number 2 is associated with the symbol "NUM" in the second place on the left-hand side, and occurs by itself in the fourth place. This means that the fourth symbol matched by the rule must be "NUM," and also that it must dominate exactly the same sub-tree as the second. In the example we are assuming that "NUM" governs a single node which will be labeled either "SG" or "PL" and the rule will ensure that whichever of these is dominated by the first occurrence of "NUM" will also be dominated by the second occurrence. Notice that noun and verb phrases could be formed simultaneously by the following rule:

```
N.1 NUM.2 V.3 2 = NOUN(1 2) VERB(3 2)
```

The symbols "ANY" and "NULL" are treated in a special way by this program and should not occur in strings to be analyzed. The use of the symbol "NULL" is illustrated in the rule:

```
PPH = NULL
```

This will cause the symbol "PPH" to be deleted from any string in which occurs. The program is nondeterministic in its treatment of rules of this kind, as elsewhere, so that it will consider analyses in which the symbol is deleted, as well as any which can be made by retaining it. The symbol "NULL" is used only on the right-hand sides of rules.

The symbol "ANY" is used only on the left-hand sides of rules and has the property that the word implies, namely that it will match any symbol in a string. The use of this special symbol is illustrated in the following rule:

```
VERB.1 ANY.1 NP.1 = VP(1)
```

This will form a verb phrase from a verb and a noun phrase, with one intervening word or phrase, whose grammatical category is irrelevant.

Elements on the left-hand sides of rules can be specified as optional by writing a dollar sign to the left or right of the symbol as in the following rules:

```
DET.1 ADJ$.1 NOUN.1 = NP(1)

VERB.1 $ANY.1 NP.1 = VP(1)
```

The first of these forms a noun phrase from a determiner and a noun, with or without an intervening adjective. The second is a new version of a rule already considered. A verb phrase is formed from a verb and a noun phrase, with or without an intervening word or phrase of some other type.

Elements can also be specified as repeatable by writing an asterisk against the symbol, as in the following example:

```
VERB.1 *NP.1 = VP(1)
```

This says that a verb phrase may consist of a verb followed by one or more noun phrases. It is often convenient to be able to specify that a given element may occur zero or more times. This is done in the obvious way by combining the dollar sign and the asterisk as in the following rule:

```
$DET.1 *$ADJ.1 N.1 *PP$.1 = NP(1)
```

According to this, a noun may constitute a noun phrase by itself. However the noun may be preceeded by a determiner and any number of adjectives, and followed by a prepositional phrase, and all of these will be embraced by the new noun phrase that is formed. Notice that the asterisk and the dollar sign can be placed before or after the symbol they refer to. The combination is often useful with symbol "ANY" in rules of the following kinds

```
N.1 NUM.2 *$ANY.3 V.4 2 = NOUN(1 2) 3 VERB(4 2)
```

This is similar to an earlier example. It combines the number morpheneme with a subject noun and with a verb, provided that the two agree, and allows for -- any number of other symbols to intervene. The symbol "ANY" with an asterisk and a dollar sign corresponds in this system to the so called variables in the familiar notation of transformational grammar.

Consider now the following rule:

```
SCONJ.1 NP(S).1 = NP(1)
```

This will form a noun phrase from a subordinating conjunction followed by a noun phrase, provided that this dominates only the symbol "S." Any symbol on the left-hand side of the rule may be followed by an expression in parentheses specifying the string of characters that this symbol must directly dominate. This expression is constructed exactly like the left-hand sides of rules. In particular, it may contain symbols followed by expressions in parentheses. The following rule will serve as an illustration of this, and of another new feature:

```
NP($DET.1 $*ANY.1 ADJ(PRPRT.2) $*ANY.3 N.4

$ PP.5)  1 3 4 WH DEF 4 BE ADJ((2)) 5
```

This rule calls for a noun phrase consisting of a noun, a preceding adjective which dominates a present participle and, optionally, a number of other elements. This noun phrase is replaced by the determiner from the original noun phrase, if there is one, the elements preceding the noun except for the present participle, the noun itself, the symbol "WH," the symbol "DEF," another copy of the noun, the symbol "BE," the symbol "ADJ" dominating exactly those elements originally dominated by "PRPRT" and, finally, any following prepositional phrases the original noun phrase may have contained. The number "2" in double parentheses following "ADJ" on the right-hand side of this rule specifies that this symbol is to dominate, not the present participle itself, but the elements, if any, that it dominates. This device turns out to have wide utility.

Double parentheses can also be used following a symbol on the left-hand side of a rule, but with a different interpretation. We have seen how single parentheses are used to specify the string immediately dominated by a given symbol. Double parantheses enclose a string which must be a proper analysis of the sub-tree dominated by the given symbol. A string is said to be a proper analysis of a sub-tree if each terminal symbol of the sub-tree is dominated by some member of the string. As usual, a symbol is taken to dominate itself. As an example of this, consider the following rule:

```
ART.1 S((ART N.2 ANY*)).1 2 = DET(1) 2
```

This rule applies to a string consisting of an article, a sentence, and a noun. The sentence must be analysable, at some level, as an article followed by a noun, followed by at least one other word or phrase. The noun in the embeded sentence, and the sub-tree it dominates, must be exactly matched by the noun corresponding to the last element on the left-hand side of the rule. The initial article and the embeded sentence will be collected as a phrase under the symbol "DET" and the final noun will be left unchanged.

The principal facilities available for writing rules have now been exemplified. Another kind of rule is also available which has a left-hand side like those already described but no equal sign or right-hand side. However it will be in the best interests of clarity to defer an explanation of how these rules are interpreted.

The user of the program may write rules in exactly the form we have described or may add information to control the order in which the rules are applied. This additional information takes the form of an expression written before the rule and separated from it by a comma. This expression, in its turn, takes one of the following forms:

n_1,

n_1/n_2,

$n_1/n_2/n_3$,

$n_1//n_3$,

n_1 in an integer which orders this rule relative to the others. Since the same integer can be assigned to more than one rule, the ordering is partial. Rules to which no number is explicitly assigned are given the number 0 by the program.

n_2 and n_3, when present, are interpreted as follows: Every symbol in the sub-string matched by the left-hand side of the rule must have been produced by a rule with number i, where $n_2 \geq i \geq n_3$. For these purposes the symbols in the original family of strings offerred for analysis are treated as though they had been produced by a rule with number 0.

2. PHRASE-STRUCTURE GRAMMAR

It will be clear from what has been said already that this program is an exceedingly powerful device capable of operating on strings and trees in a wide variety of ways. It would clearly be entirely adequate for analyzing sentences with a context-free phrase-structure grammar. But this problem has been solved before, and much more simply. We have seen how the notation can be used to write context-sensitive rules, and we should therefore expect the program to be able to analyze sentences with a context-sensitive grammar. However in the design of parsing algorithms, as elsewhere, context-sensitive grammars turn out to be surprisingly more complicated than context-free grammars.

The problem that context-sensitive grammars pose for this program can be shown with a simple example.[1] Consider the following in grammar:

$$S \rightarrow \left\{\begin{array}{l} A\ B\ C \\ D\ E\ (S) \end{array}\right\} \quad \begin{array}{l} (1) \\ (2) \end{array}$$

$$B \rightarrow \left\{\begin{array}{l} D/A\underline{\quad} \\ F/\underline{\quad}E \end{array}\right\} \quad \begin{array}{l} (3) \\ (4) \end{array}$$

$$D \rightarrow \left\{\begin{array}{l} G/A\underline{\quad} \\ B/\underline{\quad}E \end{array}\right\} \quad \begin{array}{l} (5) \\ (6) \end{array}$$

This grammar, though trivial, is well behaved in all important ways. The language generated, though regular and unambigious, is infinite.

---

[1] I am indebted for this example, as for other ideas too numerous to document individually, to Susumu Kuno of Harvard University.

Furthermore, every rule is useful for some derivation. Since the language generated is unambigious, the grammar is necessarily cycle-free, in other words, it produces no derivation in which the same line occurs more than once. Suppose, however, that the grammar is used for analysis and is presented with the string"A D E" — not a sentence of the language. The attempt to analyze this string using rules of the grammar results in a rewriting operation that begins as follows and continues indefinitely:

A D E

A B E (by rule 3)

A D E (by rule 6)

A B E (by rule 3)

etc.

It would clearly be possible, in principal, to equip the program with a procedure for detecting cycles of this sort, but the time required by such a procedure, and the complexity that it would introduce into the program as a whole, are sufficient to rule it out of all practical consideration. It might be argued that the strings which have to be analyzed in practical situations come from real texts and can be assumed to be sentences. The problem of distinguishing sentences from nonsentences is of academic interest. But, in natural languages, the assignment of words to grammatical categories is notoriously ambigious and for this problem to arise it is enough for suitably ambigious words to come together in the sentence. A sentence which would be accepted by the above grammar, but which would also give rise to cycles in the analysis, might consist of words with the following grammatical categories:

Word	Grammatical Category
1	A
2	B
3	C, E

The program, as it stands, contains no mechanism which automatically guards against cycles. However, if the user knows where they are likely to occur or discovers them as a result of his experience with the program, he can include some special rules in his grammar which will prevent them from occurring. These rules, which we have already eluded to, are formally similiar to all others except that they contain no equals sign and no right-hand side. When a P-marker is found to contain a string which matches the left-hand side of one of these rules, the program arranges that, thence forward, no other rule shall be allowed to apply to the whole string. The cycle in this latest example could not occur if the grammar contained the rule:

A B E

3. TRANSFORMATIONAL GRAMMAR

We now come to the main concern of this paper which is to discuss the extent to which the program we have been describing can be made to function as a transformational analyzer. The main purpose of the examples that have been given is to show the great power of the program as a processor of symbol strings. The notion of dominance is provided for, but only in a rudimentary way. It certainly could not be claimed that the program is a tree processor in any really workable sense. But grammatical transformations are operations on trees and our investigation therefore must take the form of showing that these operations can frequently, if not always, be mimicked by string rewriting rules.

We shall take it that a transformational grammar consists of a context-free or context-sensitive phrase-structure component and a set of transformations ordered in some way. To begin with, very little will be lost if we assume that the transformational rules are simply ordered.

Consider now the first transformation in the list. In general, this may be expected to introduce phrases into the P-markers to which it applies which could not have been generated by the phrase-structure component. Let us now write some additional phrase-structure rules capable of generating these new phrases. Let us insert these rules into the grammar immediately following the first transformational rule and establish the convention that, when they are used in the analysis of the string, their output will be used only as input to the first transformation. Now treat the second transformational rule in the same way. It also can be expected to create new kinds of phrase and phrase-structure rules can be written which would recognize these. It may be that some of the phrases formed by the second rule could also be formed by the first, and in this case, it may be possible to move the appropriate rule from its position after the first transformation to a position after the second and to mark it as providing input only for these two rules.

Notice that the rules we are proposing to construct will not constitute what has sometimes been called a surface grammar. The phrases they describe certainly do not belong to the base structure and many of them may not be capable of surviving unchanged into the surface structure. In general these rules describe phrases which can only have transititory existence somewhere in the generative process. Notice also that in order to describe these phrases adequately it may sometimes be necessary to extend the notion of phrase structure grammar somewhat. Consider for example the following transformation:

X - A - B - Y

1 2 3 4

Adjoin 2 as right daughter of 3

If we make the usual assumption that a rule is applied repeatedly until no proper analyses of the P-marker remain which can be matched by its structural index, then this transformation, and many others, may produce phrases of indefinitely many types. Let us suppose that, before this transformation is applied for the first time, all possible phrases that can be dominated by the symbol "B" are describable by context free phrase structure rules of the following form

$$B \rightarrow \left\{ \begin{matrix} \alpha_1 \\ \\ \alpha_2 \\ \vdots \\ \alpha_k \end{matrix} \right\}$$

The program, as it stands, contains no mechanism which automatically guards against cycles. However, if the user knows where they are likely to occur or discovers them as a result of his experience with the program, he can include some special rules in his grammar which will prevent them from occurring. These rules, which we have already eluded to, are formally similiar to all others except that they contain no equals sign and no right-hand side. When a P-marker is found to contain a string which matches the left-hand side of one of these rules, the program arranges that, thence forward, no other rule shall be allowed to apply to the whole string. The cycle in this latest example could not occur if the grammar contained the rule:

A B E

3. TRANSFORMATIONAL GRAMMAR

We now come to the main concern of this paper which is to discuss the extent to which the program we have been describing can be made to function as a transformational analyzer. The main purpose of the examples that have been given is to show the great power of the program as a processor of symbol strings. The notion of dominance is provided for, but only in a rudimentary way. It certainly could not be claimed that the program is a tree processor in any really workable sense. But grammatical transformations are operations on trees and our investigation therefore must take the form of showing that these operations can frequently, if not always, be mimicked by string rewriting rules.

We shall take it that a transformational grammar consists of a context-free or context-sensitive phrase-structure component and a set of transformations ordered in some way. To begin with, very little will be lost if we assume that the transformational rules are simply ordered.

Consider now the first transformation in the list. In general, this may be expected to introduce phrases into the P-markers to which it applies which could not have been generated by the phrase-structure component. Let us now write some additional phrase-structure rules capable of generating these new phrases. Let us insert these rules into the grammar immediately following the first transformational rule and establish the convention that, when they where the α_i are any strings. The phrase structure grammar needed to describe all the phrases that can exsist after the operation of this transformation must contain the following rules, or more accurately rule schemata

$$B \rightarrow \left\{ \begin{matrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{matrix} \right\} A^*$$

Where the asterisk indicates one or more repetitions of the symbol "A". If the left and right-hand sides of these rules are reversed and they are presented to the program in the proper notation, then the transformation itself can be represented by the following pair of rules:

```
B(*$ANY.1 *A.2) = 2 B+ 1 +B

B+ B.1 +B = 1
```

Since there are no facilities for specifying dominance relations among elements on the right-hand sides of these rules, it is necessary to resort to subterfuge. The phrase dominated by the symbol "B" is reproduced in the output of this rule with copies of the symbol "A" removed from the right-hand end and the remainder bounded by the symbols "B+" and"4B". These symbols serve to delimit a part of the string which can only figure in the complete analysis of the sentence if it constitutes a phrase of type "B". The second rule removes these boundry symbols from the phrase of type "B" and, since no pointer is assigned to them, they will leave no trace in the final P-marker.

Another, and perhaps more economical, way to write recognition rules corresponding to this transformation involves conflating the additional phrase-structure rules with the reverse of the transformational rule itself to give rules of the following kind:

α_i1 *A.2 = 2 B+ 1 +B ($1 \le i \le n$)

B+ B.1 + B = 1

In fact, the elementary transformation for daughter adjunction that we are providing for here is more general than that often allowed by transformational grammarians. It is common to require that if some element a is adjoined as a daughter of another element b then b must have no daughters before the transformation takes place.

Sister adjunction can be treated in an analogous manner. Consider the following transformation:

$$\begin{array}{ccccccc} X & - & A & - & B & - & Y \\ 1 & & 2 & & 3 & & 4 \end{array}$$

Adjoin 2 as right sister of 4.

The phrases exsisting before this transformation is carried out, and which have "B" as a constitutent, can be thought of as being described by a set of rules as follows:

$$\begin{array}{l} a_1 \longrightarrow B\ \alpha_1 \\ a_2 \longrightarrow B\ \alpha_2 \\ \quad\vdots \\ a_n \longrightarrow B\ \alpha_n \end{array}$$

Here the a_i are nonterminal symbols and the α_i are strings, possibly null. The grammar which describes the phrases existing after the operation of this transformation must contain, in addition, the following rules:

$$\begin{array}{l} a_1 \longrightarrow B\ \alpha_1\ A^* \\ a_2 \longrightarrow B\ \alpha_2\ A^* \\ \quad\vdots \\ a_n \longrightarrow B\ \alpha_n\ A^* \end{array}$$

The reverse transformation itself can now be represented by a set of rules as follows:

$$B.1\ \alpha_i.1\ A^*.2 = 2\ B+\ 1\ +B$$

Notice that the strings referred to by the symbols "X" and "Y" in both of the above examples are unchanged by the transformation and are therefore not mentioned at all in the analysis rules. Experience shows that it is in fact rarely necessary to write separate rules for each α_i. In most cases, a transformation of this kind could be handled in the program with a rule of the following form:

```
B.1 ANY.1 A*.2 = 2 B+ 1 +B
```

This is one of a large number of cases in which it has been found that the analysis rules can be made more permissive than the original grammar suggests without introducing spurious structures and without seriously increasing the amount of time or space used by the program.

While it is possible that transformational analysis can be done in an interesting way with a program of this sort there seems to be little hope of finding an algorithm for writing analysis rules corresponding to a given transformational grammar. The following rule also involves sister adjunction but poses much more serious problems than the previous example:

X - A - Y - B - Z

1 2 3 4 5

Adjoin 2 as right sister of 4

The problem here is that a variable "Y" intervenes between "A" and "B". On the face of it, the analysis rule corresponding to this transformation would have to be somewhat as follows:

```
*$ANY.1 B.2 *A.3 = 3 1 2
```

And in principal the program could carry out a rule of this kind. However the first symbol on the left-hand side of this rule will match any string whatsoever, so that, if the rule can be applied at all, it can be applied in a prodigious number of ways. But, with real grammars, it usually turns out that quite a lot can be said about the part of the sentence covered by the variable "Y" so that analysis rules can be written which are sufficiently specific to be practiable.

Deletions are notoriously troublesome in grammars of any kind because they can so easily give rise to cycles and undecidable problems. Transformational grammarians require that lexical items should only be deleted from a P-marker if there is some other copy of the same item which remains. This condition insures what they call the recoverability of the transformation. However, it is very important to realize that recoverability, in this sense is a very weak condition. The requirement is that, knowing that an item has been deleted from a certain position in the P-marker, it should be possible to tell what that item was. But there is no requirement that a P-marker should contain evidence that it was derived by means of a deletion transformation or of the places in it where deletions might have taken place.

Deletions are more easy to cope with in certain situations than others. Consider for example the following transformation:

X - A - B - A - Y

1 2 3 4 5

Delete 4.

The recoverability requirement is satisfied because of the identity of the second and fourth elements in the structural index. The corresponding rule for the program might be as follows:

23/22, A.1 B.2 = 1 2 1

It is necessary to provide ordering information with a rule of this kind because it would otherwise be capable of operating on its own output and cycling indefinately. But presumably this transformation can be carried out any number of times and the same therefore should be true of the corresponding analysis rule. Once again, experience shows that the grammarian almost invariably knows more about the environment in which a deletion takes place than is stated in the rule, and if this information is used carefully, analysis rules can be written which do not lead to cycles.

In principle the situation is even worse in rules of the following kind:

X - A - Y - A - Z

1 2 3 4 5

Delete 4

Here the third element is a variable which can cover any number of nodes in the P-marker. In analysis we are therefore not only without information about how many times the rule may have been applied but we know nothing about where to insert new copies of the symbol "A", except that they must be to the right of the existing copy.

The other commonly used elementary transformations (substitutions and Chomsky-adjunction) do not present special problems. The main outstanding difficulty comes from the fact that transformational rules are ordered. We have already said that the theory of transformational grammar is in the state of continual change and this is particularly true of the part that concerns the ordering of rules. For this reason we have assumed that the rules are simply ordered in the hope that other possibilities will not be notably more difficult to deal with. We shall also make the assumption that transformational rules are all obligatory.

Consider now the following grammar

Phrase structure

1. S → A (D) B C

2. C → D E

Transformations

1. A - B - X

   1 2 3

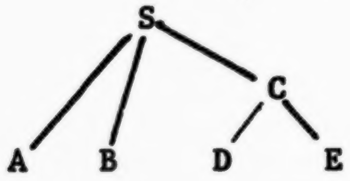
   0 2+1 3

and suppose that the program is required to analyze the string "A D B E". Since, in generation, the list of transformations is read from top to bottom it is reasonable to suppose that in analysis it should be read from bottom to top.

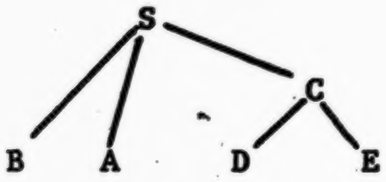
We may take it that the analysis rule corresponding to the second transformation is somewhat as follows:

D.1 B.2 = 2 1

This, together with the two phrase-structure rules, is sufficient to give a complete analysis of the string with this underlying P-marker:



But if this is an underlying P-marker, the second transformational rule could not possibly be used to produce a derived structure from it because the first transformation, which according to our assumption is obligatory, can be applied to it giving the following result:



It is in fact not sufficient to scan the list of transformations from bottom to top because this procedure does not make allowance for the fact that the transformations are obligatory. To regard transformations as optional which were intended to be obligatory is in general to associate spurious base structures to some sentences. The solution for the present grammar is to use the following set of analysis rules:

```
1/0, B D
2/1, D.1 B.2 = 2 1
3/2, A B
4/3, B.1 A.2 = 2 1
     D.1 E.1 = C(1)
     A.1 $D.1 B.1 C.1 = S(1)
```

The first and third rules contain, in effect, the structural indices from the second and first transformations respectively. The first rule says that no string is acceptable as a sentence which contains "B D" as a sub-string because to this it would have been possible to apply transformation 2. The second rule reverses the effect of transformation 2. The third rule, excludes any P-marker existing at this stage with a proper analysis containing "A B" as a sub-string. This is the structural index of transformation 1 which therefore should have been applied to any P-marker containing it. The fourth rule reverses the effect of transformation 1 and the remaining rules are the phrase-structure component of the grammar. Once again it turns out that what may be necessary in theory is only rarely needed in practice. Experience with this program is, so far, very limited but no cases have so far been found in which incorrect analysis have resulted from omitting rules such as those numbered one and three above.

CONCLUSIONS

It requires skill to write rules for analyzing natural sentences with the program described in this paper. A program can only properly be called a transformational parser if it can work directly with the unedited rules of the transformational grammar. But no algorithm is known, nor is it likely that one will shortly be found, which will produce from a transformational grammar a set of corresponding rules of the kind required by this program. It is not difficult to construct a transformational grammar for which no exactly corresponding set of analysis rules can be written. However, other programs have been written which, though they are still in many ways imperfect, can more reasonably be called transformational parsers. What then are the advantages of the present program?

The current version of the program is written in ALGOL and with very little regard for efficiency. But the basic algorithm is inherently a very great deal more efficient than any of its competitors. The various interpretations of an ambiguous sentence, or a sentence which seems likely to be ambiguous in the early stages of analysis, are all worked on simultaneously. At no stage can the program be said to be developing one interpretation of a sentence rather than another. If two interpretations differ only in some small part of the P-marker, then only one complete P-marker is stored with two versions of the ambiguous part. Work done on the unambiguous portion is done only once for both interpretations.

The program, though undoubtably very powerful, seems naive from the point of view of modern linguistic theory. The program embodies very little of what we know or believe to be true about the structure of natural languages. It might well be said that a computer program for analyzing natural languages is only interesting to the extent that it makes a claim about the basic form of those languages. But the program described here is intended as a tool and not as a linguistic hypothesis. There is much to be learned about natural language from ruminating on the form of universal generative grammar and trading counter-example for example. But there is also much to be learned from studying text as it actually occurs. The small amount of work that has so far been done with this program has been sufficient to suggest strongly that a set of rules derived algorithmically from a transformational grammar is unlikely to be the most effective or the most revealing analytic device.

11

The Relevance of Some Compiler Construction Techniques to the Description and Translation Of Languages

by

Steven I. Laszlo

Western Union Telegraph Co.*

The framework is machine-translation. Compiler-building can for a variety of reasons be considered as a special case of machine-translation. It is the purpose of this paper to explicate some techniques used in compiler-building, and to relate these to linguistic theory and to the practice of machine-translation.

The generally observed machine-translation procedure could be schematized as in FIGURE 1, or to put it another way,

1. Parsing the source-text.
2. Translation from source to object-language.
3. Synthesis of grammatically correct object-text.

FIGURE 1.

break-down, translation, and recomposition. The translation usually occurs on the level of some simplified, cannonical form (that is not necessarily the kernel-form) of both languages, such that the source-text is decomposed, and the object-text recomposed from this form. The translation algorithm usually requires a statement of the structure of both the source and the object-language, as well as the statement of some primitive-to-primitive

* Currently at Decision Systems, Inc.

correspondence paradigm for both syntactic and lexical primitives. Compilers on the other hand work on the bases of only the first two steps of FIGURE 1.: breakdown, and translation. Consequently, the processor requires only statements of the structure of the source-language and of the correspondence paradigm. That does not imply that the structure of the object-language is irrelevant to the process of translation, but that it is implicit in the correspondence paradigm, and in the selection of what is a primitive or terminal in the description of the source-language.

Through the use of examples it will be shown that BNF and similar language-description devices (8) are -- by themselves -- both analytically and generatively inadequate and depend on other devices, implicit in the translation algorithm. It will be shown that by some extensions of the notion of P-rules and some applications of the concept of T-rules (4), a description that is both analytically and generatively adequate may be constructed for programming languages. The programming language P.O.L.2 (12), (13) was selected for the examples because an adequate, fully explicit description does exist for it; furthermore, the language contains most syntactically problematic features of other programming languages as well as presenting a few unique problems in description that are worthy of attention.

The failure to come to grips with the identity problem is sufficient to demonstrate the inadequacy of BNF and similar devices (8). The simplified program-segments in FIGURE 2. serve to illustrate

EXAMPLE 1.

```
1.  Let A be variable.
2.  Let B be = "7".
3.  Let C be = "9.5".
4.  Let D be = ".072".
5.  A = B + C / D.
6.  Print A.
```

EXAMPLE 2.

```
Define Funct (A, B) = (C).
           .
           .
           .
     End.
        ---and elsewhere---
  Funct (Q, R) = (Z).
  V = D + K * Funct (P, T).
```

FIGURE 2.

this problem. BNF and similar devices would generate a parse designating "A, "B", etc. in EXAMPLE 1. as identifier (a syntactic word-class) but would fail to indicate that the various occurrences of a given identifier (e.g., "A" in statements 1., 5., and 6.) are that of the same lexical token or semantic object. Related to the identity problem is the restriction that each identifier occurring in a program statement must also occur in one and only one definition. This restriction may be called the definition problem. BNF, etc., do not handle the definition problem. Other manifestations of the identity and definition problems are associated with the use of macro- or compound functions (see EXAMPLE 2., FIGURE 2.), subscript expressions, etc.

Since there exists a demonstrable necessity for establishing the above mentioned identities and restriction (3), compilers contain -- implicit in the translation algorithm -- an elaborate table-building/table-searching/identity-testing procedure. Without such procedures, the syntactic description is inadequate, full analysis and translation impossible. In order to deal with these problems explicitly, it was decided to incorporate a transformational component along with the BNF-like phrase-structure component in the description of P.O.L.2. The above reasons for positing a transformational component are in essence the programming-language equivalents of Chomsky's original reasons to use transformations in the description of natural languages.

Rule 1. $M \to \#, M_{sel.1}, \#$

where "M" is the initial symbol, "#" is the boundary marker, and the subscript will be explained later.

Rule 2. $M_{sel.1} \to$ DEFINE, functmention, program, END

where the convention is used that terminal symbols are all capital letters, and members of the intermediate alphabet are in lower case.

Rule 3. program → . . ., placeholder, M, . . .

FIGURE 3.

In FIGURE 3., in a simplified form it is shown that the phrase-structure component generates function definitions (17), (18) embedded in others (see Rule 3.), and that the form of the function is generated in the definition -- as the expansion of the symbol "functmention" -- generating place-holders for instances of use of the function. Transformations replace the place-holders with the appropriate form of the function generated in the definition, thus accounting for both the identity and the definition problems. Other transformations exist to handle other instances of these problems e.g., labels, identifiers, subscript expressions. The method is identical: the form is generated in the relevant definition, place-holders are generated for instances of use, and the place-holders are replaced transformationally with the correct form generated in the definition.

Other transformations deal with additional notational restrictions of P.O.L.2. One such restriction is that a function definition may reference other functions but a definition may not be embedded in another. Definitions (see FIGURE 3.) are in fact generated embedded, and it becomes necessary to posit some exbedding transformation (7), moving the nested definitions outside the "parent" definition. There exist several proofs in the literature establishing the equivalence between languages generated by grammars with and without the use of boundary markers (5), (10). The exbedding transformation may be expressed more simply if boundary markers are used (see FIGURE 4.).

#, ..., #, M, #, ..., #-> #, M, #, #, ..., #

or

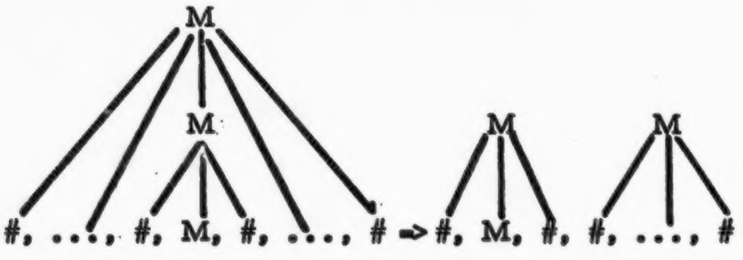


FIGURE 4.

The boundary-markers may be deleted later by another transformation, or they may rewrite as carriage-returns on some keyboard, depending on the orthography of the particular implementation and medium. The T-rules may be generated by positing a set of elementary transformations (i.e., single node operations) and a set of formation and combination rules over the set of elementary transformations, producing some set of compound or complex transformations. This is not significantly different from having locally ordered subsets of a set of elementary transformations (11), (12).

Syntactic descriptions of programming languages published in the past -- e.g., (1), (9), (19) -- generally took a program-statement to correspond to the basic unit of grammar, denoted by the initial symbol of the phrase-structure grammar. The grammer discussed here takes a function definition (see FIGURE 3.) as its basic unit. Program-statements are elements of the intermediate alphabet and have no other theoretical standing or significance. The natural language correlates of program-statements are sentences, and function definitions correspond to some larger-than-sentence units of discourse (e.g., paragraphs or chapters). This procedure may lead to some syntactic or at least linguistic method of distinguishing between "meaningful" and "meaningless" programs. Using a syntax of programs, or functions also yields an intuitively more pleasing set of relationships among elements of the described language.

The present grammar makes no effort to distinguish between "elegant" and inelegant" programming, but does distinguish both from "ungrammatical" code. Declaring arguments or variables never referenced is inelegant; referencing undeclared operands is ungrammatical. To return momentarily to the identity and definition problems: it is possible to generate a definition such that there are no corresponding place-holders; but each place-holder must be replaced by some definition-generated form of the appropriate nature. In describing the definition and use of functions, separate place-holders accomodate recursive use and the general case of usage.

It is customary to give descriptions of programming languages such that -- with the exception of some small set of key words such as arithmetic operators, delimiters of definitions, etc. -- the phrase-structure grammar generates character-strings for the lexical items. In natural languages the vocabulary is fixed. There is a stable, limited set of vocabulary elements that correspond to each syntactic word-class. In programming languages that is not the case: a small set of word-classes rewrite each as a set of one or more key-words; others will expand -- through the use of some phrase-structure rules -- as any string of characters. In the description of P.O.L.2 it was decided to separate the lexicon-generation rules from the phrase-structure rules. Though they are the same shape that BNF rules of the same purpose would be, it was determined that separating the rules generating lexical items -- even as morphophonemic rules of natural languages represent a separate class of rules -- is more intuitively acceptable: a class of orthographic rules. FIGURE 5. indicates what some of these rules might look like.

In the text of FIGURE 3., Rule 1., the explanation of the subscript was deferred. Functions and operators used in programming languages are two notational variants of the same concept (17). Depending on the notation of the system, any operation may be expressed either as an operator or a function. Since in

Rule 1. identifier → alpha ⟨, characterstring⟩

where "⟨ ... ⟩" enclose optional items.

Rule 2. $\text{characterstring} \rightarrow \begin{Bmatrix} \text{alpha} \\ \text{numeral} \end{Bmatrix} \langle, \text{characterstring}\rangle$

where "$\{\ldots\}$" enclose alternative options such that one and only one of the options enumerated must be selected.

Rule 3. $\text{alpha} \rightarrow \begin{Bmatrix} \text{A} \\ \text{B} \\ \cdot \\ \cdot \\ \cdot \\ \text{Z} \end{Bmatrix}$

Rule 4. $\text{numeral} \rightarrow \begin{Bmatrix} 1 \\ 2 \\ \cdot \\ \cdot \\ 0 \end{Bmatrix}$

FIGURE 5.

P.O.L.2 there are both functions and operators, depending on notational convenience, newly defined operations may be defined as either. Being defined as one or the other, however, restricts their distribution or "embeddability" to certain contexts. This phenomenon is accounted for by the use of a device similar to the notation of complex symbol theory (4), (11), (12), (15). The P.O.L.2 notation is such that functions (i.e., defined macros) may occur as functions, coordinate transformations (linear or otherwise) or as operands (denoting their value for a particular set of arguments) and operators may appear as arithmetic, relational or logical operators, depending on range and/or domain as well as distributional restrictions. In P.O.L.2 every program - however simple or complex -- must have an "outermost" function, one into which all others are embedded by the P-rules. The first rule of the grammar (see FIGURE 3., Rule 1.) expands the "outermost" function. Elsewhere in the phrase-structure component, depending on context, other.

"$M_{sel.1}$s" are introduced, as well as "$M_{sel.2}$s", "$M_{sel.3}$s", "$M_{sel.4}$s", and "$M_{sel.5}$s".

These correspond to the various embedded occurrences of functions and operators. The rewrites or expansions of the several versions of "M" are almost identical except for the string denoting the left bracket delimiting the definition. Alternative solutions exist but the above one appears most intuitively satisfying.

There are proofs and demonstrations in the literature to the effect that full, left, or right parenthesis notation is context-free, but not much on elided parenthesis notation. We have in the past constructed several context-sensitive grammars generating elided parenthesis notation, but they did not seem very satisfactory. Adding a device not heretofore associated with production-rules, a set of rules was produced to generate the elided parentheses notation such that the rules look and process very much like context-free rules (see FIGURE 6.).

Rule 1. $\text{expression} \rightarrow \text{expression}^{n}$

$$\text{Rule 2.}\quad \text{expression}^{n} \rightarrow \left\{ \begin{array}{l} \text{expression}^{n+e}\text{, operator}^{n}\text{, expression}^{n} \\ \text{expression}^{n+e} \\ \text{"(", expression, ")"} \\ \text{identifier placeholder} \\ \text{unaryoperator, expression} \end{array} \right\}$$

where for one cycle (11) $\underline{n}$ remains the same integer between subrules 1 and 2 and $\underline{e}$ remains the same integer increment.

FIGURE 6.

Though the "counter" $\underline{n}$ and the "increment" $\underline{e}$ are not part of a known system of production rules, their nature and the reason for their use can be clearly stated. Their use permits a simpler scanner for the syntax than context-restricted rules do.

A similar counter is used to handle the concatenations of n-tuples.. In P. O. L. 2 an item of data may be declared as a pair, triple, or n-tuple, and operations may be performed over n-tuples of identical $\underline{n}$-s (see FIGURE 7.).

Rule 1. n-tuple-expression → n-tuple, operator, n-tuple

where n = n = n. Any of the n- tuples may however be concatenates of two or more n-tuples of smaller n-s such that:

Rule 2. n-tuple → (m) - tuple, concatenator, (n-m)-tuple

where n and m are positive integers and the arithmetic relationship designated obtains.

FIGURE 7.

Of course, the (m)-tuple or the (n-m)-tuple may be further broken down by the same rule into further concatenates.

The above are selected examples rather than an exhaustive list of the transformations in the syntax of P.O.L.2. A rigorous statement of the transformations is available, stated as mappings of structural descriptions into structural descriptions, accounting for the attachment and detachment of nodes. Presenting the selection of transformations here in a descriptive rather than a rigorous form offers an idea of the general approach.

Constructing the phrase structure component, many alternative solutions or approaches came up at every juncture; in specifying the transformational component, the alternatives quickly multiplied beyond manageable proportions. It is certainly the case that throughout its brief but exciting history, one of the aims of transformational theory has been to describe language in terms of the most restricted -- hence simplest -- system possible. But one may well regard the sets of devices sofar advanced as parts of transformational theory, as algorithmic alphabets (in the A.A. Markov/Martin Davis (5), (15) sense). Specific algorithmic alphabets are more or less arbitrary selections from some universe of elementary and compound algorithms bound by formation and combination rules. This paper is not a proposal toward the modification, extension or restriction of transformational theory, merely an indication that an overlapping set of algorithms may be selected to deal with a similar but not identical problem: the structural description of some formal notation systems such as programming languages.

Beyond doubt, substantial simplification and sophistication may be achieved over the model described here. The effort here has been toward the application of linguistic techniques to artificial languages, conforming to the linguist's notion of what it means to "give an account of the data", rather than to the laxer standards of the methods used to describe programming languages.

BIBLIOGRAPHY:

1. Backus, J.W. "The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference", Information Processing; Proceedings of the International Conference on Information Processing. Paris: UNESCO, 1960.
2. Cheatham, Jr., T.E. The Introduction of Definitional Facilities into Higher Level Programming Languages. Draft Report CA-6605-0611.; Wakefield, Mass.: Computer Associates, Inc., 1966.
3. ________________ The Theory and Construction of Compilers Draft Report CA-6606-0111.; Wakefield, Mass.: Computer Associates, Inc., 1966.
4. Chomsky, Noam. Aspects of the Theory of Syntax. Cambridge, Mass.: MIT Press, 1965.
5. ________________ "On Certain Formal Properties of Grammars", Information and Control, 2, (1959), pp. 137-167.
6. Davis, Martin. Computability and Unsolvability. New York; McGraw-Hill, 1958.
7. Filmore. C.J. "The Position of Embedding Transformations in a Grammar", Word, 19, 2, (1963).
8. Gorn, Saul. "Specification Languages for Mechanical Languages and their Processors -- A Baker's Dozen", Communications of the ACM, 7, 12, (1961).
9. Heising, W.P. "History and Summary of FORTRAN Standardization Development for the ASA", Communications of the ACM, 7, 10, (1964).
10. Landweber, P.S. "Three Theorems on Phrase Structure Grammars of Type 1", Information and Control, 6, (1963), pp. 131-136.
11. Lakoff, G.P. Cycles and Complex Symbols in English Syntax. Unpublished Manuscript, Indiana University, 1963.
12. ________________ Some Constraints On Transformations. Unpublished manuscript, Indiana University, 1964.
13. Laszlo, S.I. "Report on a Proposed General Purpose Procedure Oriented Computer Programming Language". Report of the Institute of Educational Research, Bloomington: Indiana University, 1965.
14. ________________ "P.O.L., A General Purpose, Procedure Oriented Computer Programming Language" Report of the Institute of Educational Research, Bloomington: Indianan University, 1965.
15. Matthews, P.H., "Problems of Selection in Transformational Grammar", Journal of Linguistics, 1, (1965).
16. Markov, A.A., Theory of Algorithms. Washington, D.C.: U.S. Printing Office, 1965.
17. McCarthy, John "A Basis for a Mathematical Theory of Computation", in Computer Programming and Formal Systems. P. Braffort & D. Hirschberg (ed.), Amsterdam: N. Holland

Publishing Co. 1963.

18. ____________ et al., LISP 1.5 Programmer's Manual; Cambridge, Mass.: MIT Press, 1962.

19. Naur, Peter (ed.) "Revised Report on the Algorithmic Language ALGOL 60", Communications of the ACM, reprinted in E.W. Dijkstra, A Primer of ALGOL Programming. New York: Academic Press, 1964.

12

Conférence Internationale sur le traitement
Automatique des Langues.

- - - - - - - - - - - - -

MAN-AIDED COMPUTER TRANSLATION FROM ENGLISH INTO FRENCH USING AN ON-LINE SYSTEM TO MANIPULATE A BI-LINGUAL CONCEPTUAL DICTIONARY, OR THESAURUS.

Author: MARGARET MASTERMAN

Cambridge Language Research Unit,
20, Millington Road,
CAMBRIDGE.
ENGLAND.

Conférence Internationale sur le traitement
Automatique des Langues.

Author: Margaret Masterman

Institute: Cambridge Language Research Unit,
20, Millington Road, CAMBRIDGE, ENGLAND.

Title: MAN-AIDED COMPUTER TRANSLATION FROM ENGLISH INTO FRENCH USING AN ON-LINE SYSTEM TO MANIPULATE A BI-LINGUAL CONCEPTUAL DICTIONARY, OR THESAURUS.

Work supported by: Canadian National Research Council.
Office of Naval Research,
Washington, D.C.

background basic research earlier supported by:
National Science Foundation,
Washington, D.C.
Air Force Office of Scientific
Research, Washington, D.C.
Office of Scientific and Technical
Information, State House, High
Holborn, London.

[N.B. References will be denoted thus: ◯]

I. Long-term querying of the current state of despondency with regard to the prospects of Mechanical Translation.

The immediate effect of the recently issued Report on Computers in Translation and Linguistics, LANGUAGE AND MACHINES ① has been to spread the view that there is no future at all for research in Mechanical Translation as such; a view which contrasts sharply with the earlier, euphoric view that (now that disc-files provide computers with indefinitely large memory-systems which can be quickly searched by random-access procedures) the Mechanical Translation research problem was all but "solved".
It is possible, however, that this second, ultra-despondent view is as exaggerated as the first one was; all the more so as the Report is written from a very narrow research background without any indication of this narrowness being given. For example, an M.T. Thesaurus has never yet been put on a machine; ② and the analogy between M.T. and Information-Retrieval has never yet been explored, (yet retrieving a translation in response to a user's request is basically the same as retrieving any other piece of information in response to a user's request.) ③ No mention, moreover, is made in the Report of the work of

(e.g.) Dolby and Resnikoff in analysing the nature and structure of natural-language dictionaries, ④ nor is any recommendation made that more of this evidently necessary work should be done. Moreover, the need for basic research into the true problem posed by the ambiguity and extensibility of individual language-signals of any order of length, and the connection of this with other learning-problems and character-recognition-problems, has never yet been faced. In fact, the situation is worse; a particular application has been pronounced useless and/or impossible before the general field of examining the basic semantic nature of human communication has been created.

II. Recommendation: do not look at the theoretic complexities of current researches into language-problems: look rather at the technological advances which have already been made.

Thus the basic recommendation given in the Report, namely that practical research into Mechanical Translation should be discontinued, while present, very narrow and fragmentary trends of "pure" theoretic linguistics research should be supported, can be queried both ways round. For the advances in this field are precisely coming from the technologies, as the Report itself shows, and that in several areas i) Thus computer-type-setting, in which hyphenation can be done with a "logic", that is, without a dictionary, is now an accomplished fact ⑤ ii) within information retrieval, mechanized retrieval systems of increasing sophistication and efficiency, are being constructed for practical use at Universities and within industry: iii) synthetic speech considered as synthetic message, - passed over in the Report because created by telephone engineers and not by linguists, - is making great strides ahead; iv) high-level programming languages increasingly operate more like natural languages, so that the machine can pick up and process something more like the user's normal way of thinking; v) the Mannheim and Luxembourg machine-aided translation-systems are acknowledged in the Report to save 40 - 60 per cent of a translator's time; ⑥ and vi) research in automatic character-recognition has now reached such a point that consideration of the extent to which this will slash M.T. costs and increase M.T. usefulness should have not been ignored. ⑦

III. Report on an actual experiment in man-aided M.T.

The experimental work to be reported on in this paper and which is still in progress, is the development of a computer-aided procedure for the full translation of *one single paragraph* of governmental report-style English into governmental-report-style Canadian French, to be made in such a way that the translation *actually* produced accounts for the *actual* non-literal translation which was *actually* made by the official Canadian Government Translator.

The philosophy behind this research is that before employing automatic-translation-devices on a large scale, you have got to understand what translation is yourself; just as before building a liner-smoke-funnel you have got to understand wind-flow. You may not in the end use, to assist translation, all the mechanical procedures which you develop in order to understand translation, but you have got to know what these are; mechanically speaking, you have not got to be continually surprised and taken aback by what the human translator actually does.

Even the amount of experimentation which we have performed so far has sufficed to convince us that nobody does know, in terms of automatic procedures, what translation is. So-called M.T. programs, up to now, though they have performed ~~only~~ more or less sophisticated feats in bi-lingual transformation of individual words and of individual constructions, have never in the true sense of the word, translated anything.

We have now, however, started to put on a machine a more realistic translation-model of the following form. The model draws on ii) iii) iv) and v) of the technological devices mentioned above. i) As is standard practise now on Information Retrieval, the model uses a Thesaurus. This Thesaurus, however, is not merely an Information-Retrieval-type Thesaurus of terms, but a "Roget's Thesaurus" type of technical dictionary, though of a novel kind. ii) The retrieval-procedure works by using as its "requests" a unit longer than the word, and which has been called a "phrasing" (*Fr: phrase rhythmique*); ⑧ a computer-program, (written by J. Dobson for the Titan Computer at Cambridge University Mathematical Laboratory) now exists which derives phrasings from written text (see appendix A) iii). The user is on-line to a computer, on which the whole Thesaurus is stored; and he reacts with this Thesaurus by means of question-and-answer routines operating in real time which are programmed into the machine by using the very sophisticated programming language T.R.A.C. ⑨. And v), the experiment presupposes the validity of the result that, in operation, the computer-stored dictionaries at Luxembourg and Trier (to which the user is not on-line and with which he cannot therefore react,) already, in spite of these limitations save 40-60% of the translators' time. It is inferred from this that on-line use of more sophisticated dictionaries by man-machine interaction in the conversational mode is the right way, from now on, for M.T. research to go.

III. The Basic Principle of the Man-Machine interaction.

The input to the machine is a stressed and contoured phrasing, i.e. a phrasing with some stresses marked and minimal syntactic naming of the constituent words. Research to produce this input mechanically, by a phrasing-stresser-and-parser, is currently being supported by the Office of Scientific and Technical Information, London; at present the program (Mark II) segments the text into phrasings mechanically, but does not either mark the stressed words or provide any snytactic naming. (see Appendix A). In the mini-demonstation of the man-mahhine interaction, therefore, (the only one which is already operational as a machine,) the operator at present types in a single phrasing at a time minus the stressed words, which have been pre-marked on his text. Thus, he does not type in a complete phrasing, but what we have called a phrasing-frame. (Later the machine will compute the phrasing-frame from the text.) Examples of assorted phrasing-frames are given below:

ASSORTED PHRASING-FRAMES

HE..........ED TO THE..........
[PAST VERB] [NOUN]

THERE IS A.......... IN THE..........
[NOUN] [NOUN]

HE WENT ▲ TO THE..........
[NOUN]

.... ()
[ABSTRACT NOUN]

....
[ADJ] [NOUN]

ON...................... ()
[ABSTRACT NOUN]

ON................... ....
[ADJ] [NOUN]

ANY..▲.......... ()
[NOUN]

A.............. ....
[ADJ] [NOUN]

SUCH AS............. ()
[NOUN]

SUCH AS............. IN...........
[PARTICIPLE] [NOUN]

MUST BE PARTICULARLY ▲...........
[ADJ]

TO................ ITS...........
[VERB INFIN.] [NOUN]

key:	stressed word omitted
()	silent beat
▲	do not translate though stressed.

N.B. Other markers e.g. the marker ■ to set in operation a routine to inter-connect syntactically connected phrasings will be discussed in a further publication

On receiving the phrasing-frame, the machine questions the operator in order to make him specify further, from his general knowledge of the text and of its subject, what the context of the particular phrasing-frame is. The example given below, in which is progressively specified the correct French translation of an English verb of motion (one of the notoriously difficult English forms to translate into French) shows how complicated this questioning can be. Not more than three rounds of questioning are allowed, and when the operator has produced his specification, the unique correct translation of the frame is stored in the immediate-access store of the machine (see Appendix B). In the example set out below, however, the differing French translations of all possible answers obtainable under Round II and Round III of the inter-action are set out immediately underneath the English statements which the machine would actually print out on the console, in order to show the underlying reason for the whole enterprise.

STAGE I:

ROUND I

Operator types in the phrasing-frame:

HE XX-ed TO THE XX
[PAST VERB] [NOUN]

Machine asks: DO YOU MEAN

A HE MOVED TO SOME POINT ?

B HE DETERIORATED (idiom)?

C HE CROSSED OVER SOME OBSTACLE ?

D HE COMMUNICATED WITH SOME BEING ?

Operator answers: A

ROUND II

Machine then asks: DO YOU MEAN

A HE DEPARTED TO SOME DESTINATION ?
Il partit pour ...

B HE TRAVELLED TO SOME DISTANT REGION ?
Il s'en alla à ...

C HE FLED TO SOME REFUGE ?
Il se réfugia chez ...

D HE ARRIVED AT SOME DESTINATION
Il arriva à ...

E HE VISITED SOME PLACE ?
Il visita ...

Operator answers: B

ROUND III

Machine then asks: HOW DID HE TRAVEL?

A BY PLANE ?
Il prit l'avion pour ...

B BY SEA ?
Il voyagea par bateau à ...

C BY ROAD ?
Il voyagea en voiture à ...

D BY TRAIN ?
Il prit le train pour ...

E ON FOOT ?
Il se rendit à pied à ...

F BY BICYCLE ?
Il s'en alla à bicyclette à ...

G BY SWIMMING ?
Il alla à la nage à ...

Operator answers: A

STAGE TWO

The operator then types in the two stressed words:

FLEW and FRONTIER

The machine then dictionary-matches and resolves:

FLEW = XX-ed = ALREADY TRANSLATED: DELETE

XX = FRONTIER = FRONTIÈRE (f)

and immediately, for the text:

He flew to the frontier

The Machine prints out the translation:

IL PRIT L'AVION POUR LA FRONTIÈRE

— — — — — — — — — o — — — — — — — — — —

Detailed examination of this example shows that behind this particular way of making an on-line system interact with an operator there lies a *strategy*, a *hypothesis* and a *prospect*.

V. *The strategy* is at all costs to avoid post-editing; but to allow maximal pre-processing of the input text by the machine interacting with the *operator, all the question-and-answer routines being in the operator's native language*.

The argument against post-editing (as the U.S. *Report* conclusively shows) is that it is either mechanical - e.g. the resolution of French gender-concord - in which case the machine itself can be programmed to do it - or it is creative and/or intuitive, in which case it cannot be done at all without extensive reference back to the input text. (Who could interpret "Shakespeare Overspat", which was the title of a Russian "Pravda" article as translated by the U.S. Air Force computer? The real meaning was "Shakespeare is now a back number"), in which case the post-editor might as well have translated the whole text himself in the first place.

To avoid post-editing, however, the output produced by a man-machine reactive M.T. program has either got to be a blank space (when the program fails), or a unique translation which is known to be correct. Now uniqueness of output can be brutally produced, as everybody knows, by programming the machine only to print out one of any set of alternatives. Correctness, however, can only be achieved by the target-language translation having been approved beforehand by the operator, from cues which the machine gives him, or which he gives the machine - *in his own language*; i.e. in the source language. The real use, therefore, of the three-stage question-and-answer routine exemplified above, is that it enables an Englishman with a console but who does not know any French to produce a unique and correct idiomatic French translation of an English text, provided that he is prepared to take the trouble to pre-process the English text so that it is finally restated in a Frenchified sort of way. After this the machine can of course transcribe it into French.

In other words, a machine-aided translation program basically consists -

a) of programming the machine to pick up the ambiguities in the source language which the target-language will not tolerate (not the other way round) and of making the operator produce the additional information which will resolve them.

Take, as example, the phrasing

/for a standby force/.

This looks technical and unambiguous in the English, but comparative examination of bi-lingual text showed that it translated into French (and in the same document) as either

i) /d'une force d'urgence/, i.e. /"of an emergency force"/
or ii) /pour une force de réserve/ i.e. /"for a reserve force"/,

according to sophisticated considerations of context. Therefore, when the operator types the technical term STANDBY FORCE into the machine, in order to fill up the gaps in the phrasing-frame /FOR A.......... ....
[Adj] [Noun]

the machine has got to answer him back:

DO YOU MEAN

A AN EMERGENCY FORCE

B A RESERVE FORCE

The operator then has to choose, and type back into the machine the alternative he wants, after which the machine can make the translation.

b) Similarly, a way must be found of enabling the machine to pick up, from cues in the source language, the metaphors and idioms which the target-language will not tolerate, and to assist the operator to rephrase the stretch of text concerned, in terms which the target-language will tolerate. The difference between idioms and metaphors is that idioms can be mechanically picked up and matched by an idiom dictionary, whereas metaphors can't.

c) Similarly again, the machine must be programmed to pick up, from the source language input, the constructions which the target-language will not tolerate, and assist the operator to transform these into constructions which the target-language will tolerate (e.g. to turn English passives into French actives, and the adjectives of English adjective-noun strings into French post-positioned prepositional phrases).

Thus the whole translating work, really, is done within the source language. Once you can preprocess your English input into a Frenchified shape in the respects a), b), c), above, the machine can transform this Frenchified English, with no trouble at all, into elegant French.

The strategic hope, of course, is that by analysing the printouts produced by a large number of sequences of such machine-man interactions, in translating many types of texts, we shall ultimately learn how to make the machine answer, as well as ask, some of the rounds of questions, (as is already being done in a whole range of machine "edit" programs), so that the machine shall progressively become able to do more of the Frenchification process for itself; thus finally producing, (if the machine ever became able completely to take over) exceedingly slow but reliable machine translation, - which could, subsequently again, be speeded up.

Before further discussion of the extent to which this strategic hope is a real hope and how much a mere pious aspiration, i.e. the prospect, I will now set out the hypothesis (as opposed to the strategy) of the experiment.

VI. The hypothesis which the translation-model gives is the following:

Translation consists of the pairing of a phrasing, P_7^A, in Language A, with another phrasing, P_2^B, in Language B, in such a way that P_1^B ~~[illegible]~~ forms an analogy with P_1^A, in a sense of "analogy" which can be ostensively defined intterms of the model. Thus translating a phrasing into another language is no different, (according to this translation-model) from defining it, producing a parallel-phrasing to it, reiterating or otherwise further specifying it, in the same language.

The advantage of the model is that unambiguous criteria of the formation of such a pairing can be given. For any response given by the operator to a machine-question will form such a pair: the first member of the pair will be the original phrasing, (in English), the second the chosen machine-specification (called by us a template) also in English. Then another pair will be formed whenever the machine translates the operator's final choice of template into French; the first member of the pair in this case, will be the final template chosen, and the second member will be the translation into French, with the stressed words translated and inserted into their correct places. Then again, an intermediate pair may be formed of which each member is a template; the first member of such a pair will be a more abstract template chosen at the first round of man-machine interaction, while the second member of it will be the more concrete template chosen by the operator at the second round of man-machine interaction; and so on recursively.

Any such pairing formed by the translation model, whether between English phrasing and template, or between template and template, or between template and French phrasing, we shall call a semantic square. A philosophic discussion of the notion of semantic square is given in another publication (10).

A semantic square (in terms of this model) consists of the pairing of any two linguistic sequences P_1 and P_2, P_1 and P_2 each having the following characteristics:

i) each has two stressed segments (which when P_1 is paired to P_2, form points of the square).

ii) each has these embedded in some phrasing-frame, (which, when P_1 is paired to P_2 forms the frame of the square).

iii) each has been selected as synonymous with the other at least once, either by the operator or by the machine.

Thus, according to the model, translation consists of sequential semantic-square forming, the sequence of semantic squares thus formed continuing until it is brought to an end by the machine printing out a square which has a target-language phrasing as its second member.

To make all this clearer, let us further develop the example of man-machine interaction given above, by assuming that the phrasing to be translated is

/HE WENT to the police/.

To translate this, the operator types in

/HE....ED to the......../
[PAST VERB]   [NOUN]

and chooses, at the first round of questioning, the abstract template

HE COMMUNICATED WITH SOME ANIMATE BEING

The first semantic aquare of this sequence formed by the model is thus:

/HE WENT TO THE POLICE/

/HE COMMUNICATED WITH SOME ANIMATE BEING/.

The machine then asks: DO YOU MEAN

A HE REVEALED-ALL TO THE ENEMY

B HE TOLD-A-STORY TO SOME LISTENER

C HE CONSULTED WITH SOME AUTHORITY

The operator chooses A, thus forming the second semantic square in the sequence:

/HE COMMUNICATED WITH SOME ANIMATE-BEING/

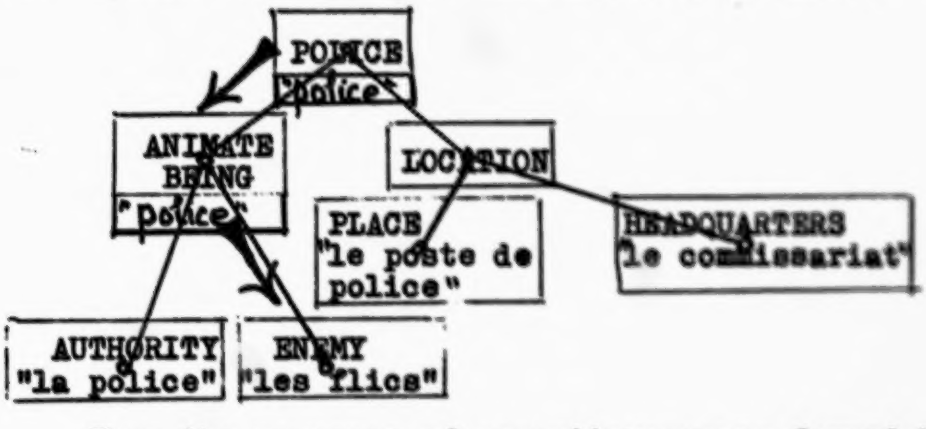
/HE REVEALED-ALL TO THE ENEMY/

The operator then types in the stressed word /POLICE/ (to specify the nature of the enemy), and the machine then forms the final semantic-square:

/HE REVEALED-ALL TO THE ENEMY/

/IL TOUT RÉVÉLA AUX FLICS/

"FLICS" having been pre-chosen by the operator's choices of template from a bi-lingual tree-dictionary-entry for the English word "police" with nodes as follows:


POLICE
"police"
ANIMATE BEING
"police"
LOCATION
PLACE
"le poste de police"
HEADQUARTERS
"le commissariat"
AUTHORITY
"la police"
ENEMY
"les flics"

Thus the sequence of semantic-squares formed by this operation of the model is

1 HE WENT TO THE POLICE
HE COMMUNICATED WITH SOME ANIMATE-BEING

2 HE COMMUNICATED WITH SOME ANIMATE BEING
HE REVEALED-ALL TO THE ENEMY

3 HE REVEALED-ALL TO THE ENEMY
IL TOUT REVELA AUX FLICS

This square-sequence, with its AB BC CD overlap of content, I will call the semantic deep-structure of the model's translation-operation, and the tree-structure given above I will call the semantic deep-structure of the dictionary-entry.

The totality of semantic deep-structures given by the model is the model's total semantic-field.

VII This, stated in the briefest possible terms, is the hypothesis given by the model. Now as to the prospect of developing this line of research.

The first thing to say is that the model makes clear the unsuitability of the ordinary digital computer as compared to a human being for performing translation. For in this translation-model the computer handles each phrasing of the input text as a separate unit, and forces the operator, by successive rounds of questioning, so to specify it that it can be translated unambiguously into French. But the human being, who does not treat each phrasing of a text as a separate unit, but who uses his understanding of the earlier phrasings of a text to guide him in his understanding of the later ones, does not have to ask himself nearly so many questions. A progressive learning-model of translation, then, is what is really required, rather than the present single-phrasing-matching model. On the other hand, the complexity which has to be introduced into the model to account for all the differing French translations which have to be made of a single piece of English, according to its context, this would have to be introduced into any effective M.T. program: since you cannot retrieve from any computerised data-system any data which you have not first put in. But this second type of complexity can be put into the machine gradually, by feeding in data obtained from examining the inter-lingual correspondences in a large corpus of bi-lingual text.

There is, however, another, much deeper obstacle to developing this research, and that is that (as M.T. research-workers have for some time past suspected) bi-lingual dictionaries provide almost no clue to semantic deep-structure.

Within the context of the present experiment this became apparent in examining the English word "deliberations". The examination began with the construction of a dictionary-entry-card of the following form:

English: DELIBERATIONS
French: DELIBÉRATIONS

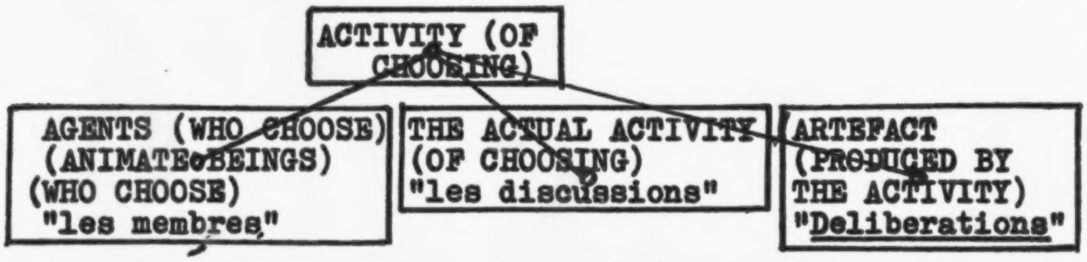
This entry being queried (and the maker of it having defended himself by saying that "deliberations" was the only word he knew of in English which could really be translated by the corresponding word in French), it was checked with Vinay's Dictionary(1) which gave the entry /débats mpl, discussion/. However, when an investigation was made of how it was actually translated in the corpus of text, it only occurred once, where it was translated "membres", as follows:

English
The illustrative and comparative
materials presented
may prove helpful
to the deliberations of this committee

French
Les données explicatives
et comparatives ()
se révélerent, peut-etre
très utiles
pour les membres du comité

Moreover, the translator, in translating it thus, was quite right; not only because "utiles" in French, likes a concrete complement, but also because this is what the passage means.

However, this dictates a semantic deep-structure for the bi-lingual dictionary-entry of "deliberations" of the following form:


ACTIVITY (OF CHOOSING)
AGENTS (WHO CHOOSE) (ANIMATE BEINGS) (WHO CHOOSE) "les membres"
THE ACTUAL ACTIVITY (OF CHOOSING) "les discussions"
ARTEFACT (PRODUCED BY THE ACTIVITY) "Deliberations"

It becomes evident, then, that if we are to make a machine account for the translations, which good human translators actually produce, using the kind of model which has been reported on in this paper, the problem is that of finding the ~~output~~ deep structures of the dictionary-entries from the data actually given by a bilingual corpus; for the construction of the square-forming templates must depend on these- that is if the template-glossary and the bi-lingual dictionary are to interlock.

Present research efforts are therefore being concentrated on the problem of "forming up" the whole notion of semantic dictionary-entry deep-structure.

CONCLUSION

In view of the great interest which has already been aroused by this experiment, its small scale and pilot nature must be emphasized. (Actual output from a trial run of the program is given in Appendix B). It has been implemented only on an I.C.T. 1202 computer, with T.R.A.C. facility, to which a single keyboard has been attached, just under the print-out, on which the machine's "replies" to the operator, as well as his "questions" appear. This machine has only 4K store with no back-up, and 2K of this is occupied by the T.R.A.C. facility; the rest of the store will therefore only hold enough Thesaurus to process an average of 10 "phrasing-frames" at any one time, so the sections of Thesaurus which are needed for any particular test have to be prechosen by hand from the larger deck of punched cards of which the Thesaurus, in its machine-readable form, consists. Even these cards, however, are only punched as required; the basic triple dictionary, from which the Thesaurus is being built up, is being stored on ordinary business equipment, (Twinlock Handifed Binder HRA 3 handled with a Shunic Signalling System and Paper and a SASCO System so as to ensure maximum flexibility and ease of entry-change).

Mark II of this program is to be implemented on an I.C.T. 1903 with disc-file and multiple-access T.R.A.C. facility, but this is not expected to be operational till 1968.

REFERENCES:

1. Language and Machines: Computers in Translation and Linguistics. Available from Printing & Publishing Office, National Academy of Sciences, 2101 Constitution Avenue, Washington, D.C. 20418. Price $4.00.

2. A project supported by the U.S. National Science Foundation at the University of Bloomington, Indiana, has just been started, to make a Thesaurus for Information Retrieval in 50 languages.

   Also a historical Thesaurus of English is being compiled on a long-term basis by Professor Samuel at the University of Glasgow; and another, compiled by John Bromwich, is being put on magnetic tape at the Linguistics Computation Centre, Cambridge University.
   The properties and structure of thesauruses and/or conceptual dictionaries have never yet, however, been mechanically examined; partly because, until lately, machines with rapid-access-time to sufficiently large memories were not available, and partly because of the overall cost of such a project.

3. Margaret Masterman, R.M. Needham & K. Sparck-Jones: The Analogy between Mechanical Translation and Library Retrieval, (Proceedings of the International Conference on Scientific Information, 1958), Washington, D.C., National Academy of Sciences, 1959, p.917.
   See also, on this analogy,
   (i) Margaret Masterman: Translation, (Proceedings of the Aristotelian Society, 1959-60, p.79);
   (ii) R.M. Needham & I. Joyce: The Thesaurus Approach to Information Retrieval, (American Documentation, Vol. 9, 1958, p.192).

4. J.L. Dolby & H. Resnikoff: The English Word Speculum in 5 vols., (Lockheed Missiles & Space Company, Sunnyvale, Cal.) 1964.
   On the Structure of Written English Words (Language Vol. 40, No. 2,) 1964.

5. Language & Machines, p.114. The Report gives this brilliant technical achievement just 3 sentences on p.114, and appears not to know of the fact that a mechanical justifier using a logic and working up to 95% accuracy is now in use on an actual newspaper (personal communication from Dolby & Resnikoff).

REFERENCES (cont.)

6. *Language & Machines*, p.26.
See also
i) F. Krollmann, H.J. Schuck and U. Winkler: *Production of Text-related Technical Glossaries by Digital Computer*, (mimeo, undated);
ii) *La Terminologie*, *Problemes de Coopération Internationale*,
Expose de M.J.A. Bachrach, Chef du Bureau de Terminologie de la Haute Autorite de la C.E.C.A. a Luxembourg - (The Applied Linguistics Foundation) a Strasbourg - Maison de l'Europe, le 6 Septembre, 1965. (mimeo).
iii) Lydia Hirschberg: *Dictionnaires automatiques pour Traducteurs humains*, (Journal des Traducteurs, Montreal, Vol. 10, No. 3 (1965), pp. 78-86.
iv) Lydia Hirschberg: *Dictionnaires Automatiques Multilingues, Conception, Utilisation, Realisation*, (Colloque sur la Terminologie, Luxembourg, 1er avril, 1966. Université Libre de Bruxelles, Centre de Linguistique automatique appliquée). (mimeo).

7. *Language & the Machine*, pp. 32-33.

8. D. Shillan: *Spoken English*, Longmans, Green (London) 1954/65;
D. Shillan: article in *META* (Montreal), Vol. XI, No. 3, 1966;
D. Shillan: article in *English Language Teaching* (Oxford), Vol. XXI, No. 2, 1967.
(See "Segmenting Natural Language by Articulatory Features" in the present Conference.)
The phrasing method offers two operational simplifications
i) by mapping the distribution of stresses on to a binary frame;
ii) by applying a phonetically-derived feature to *words*, instead of to syllables or phonemes.

9. Calvin Mooers: *T.R.A.C., A Procedure Describing Language for the Reactive Typewriter* (Vol.9 No.3 1966, Communications of the A.C.M.)
R. McKinnon-Wood,& D.S. Linney: *T.R.A.C*. (Vol.2 of Report to O.S.T.I. on Automatic Syntax 1966)

10. Margaret Masterman: *Semantic Algorithms* (Las Vegas Conference on Computer-related Semantics, 1965)

11. *Everyman's French-English English-French Dictionary with special reference to Canada,* compiled by Jean-Paul Vinay, Pierre Daviault, Henry Alexander, (Dent & Sons, 1962) p.494.

APPENDIX A (a)

```
JOB TITLE - (JED744/PHRASING SORT)...24 5 67
STREAM 1/0   INITIAL INPUT

0/1/1          *LIMITATIONS

0/1/2          ON *CANADIAN *COMMITMENTS.

1/1/1          *ANY *NATION

1/1/2          *MUST BE *CONCERNED

1/1/3          THAT ITS *OBLIGATIONS

1/1/4          DO NOT *OUTRUN ITS *CAPABILITIES.

1/2/1          A *MIDDLE *POWER

1/2/2          SUCH AS *CANADA

1/2/3          MUST BE *PARTICULARLY *CAREFUL

1/2/4          TO *RATION ITS *COMMITMENTS.

1/3/1          *ALTHOUGH AT THE *END+OF+THE+WAR

1/3/2          *CANADA *COULD+HAVE+DEVELOPED

1/3/3          THE *CAPABILITY

1/3/4          TO *MANUFACTURE *NUCLEAR+WEAPONS

1/3/5          IT *ELECTED

1/3/6          AS A MATTER OF *DELIBERATE *CHOICE

1/3/7          *NOT TO *BECOME

1/3/8          A *NUCLEAR *POWER.

2/1/1          *ALSO, *CANADA

2/1/2          DID *NOT *BECOME+A+PARTY

2/1/3          TO THE *INTER *AMERICAN

2/1/4          *DEFENCE *SYSTEM.

2/2/1          *AND, AT THE *CONCLUSION

2/2/2          OF THE *KOREAN *WAR

2/2/3          *CANADA *WITHDREW

2/2/4          HER *TROOPS FROM *THAT+AREA.
```

APPENDIX A (b)

2/3/1 *SUBSEQUENTLY, *CANADA
2/3/2 *DID+NOT *ASSUME

2/3/3 *REGIONAL+DEFENCE+OBLIGATIONS IN THE *PACI
2/3/4 SUCH AS *PARTICIPATION IN *SEATO.
2/4/1 IN *CONSEQUENCE
2/4/2 *LIMITS HAVE BEEN *SET
2/4/3 TO OUR *MILITARY *RESPONSIBILITIES.

APPENDIX B (a)

```
          THERE IS A FLAW IN THE PROOF

          10

          11

          V10CL&TRAN11

     START

          THERE IS A XX IN THE XX

     START
```

APPENDIX B (b)

```
                PROOF

       START

                    DO YOU MEAN
                    A   DEMONSTRATION
                    B   TYPOGRAPHIC PROOF


                A

       START

                    THERE IS A FLAW IN THE PROOF
       END          IL SE TROUVE UN 7 UNE ERREUR DANS CE 7 CETTE PREUVE
```

APPENDIX B (c)

```
                DO YOU MEAN
                A   CONCRETE OBJECT IN ENCLOSURE
                B   CONCRETE OBJECT IN LOCATION
                C   EVENT IN FUTURE TIME
                D   ERROR IN ARGUMENT


            D


     START


            FLAW


     START
```

13

RESUME

Johannes Meyer-Ingwersen
2 HAMBURG 13
Von Melle Park 6
Orientalisches Seminar

ANALYSE DE PROPOSITIONS ALLEMANDES

(Procédures spéciales après avoir trouvé la racine du verbe /geh-/)

Le sujet de ma communication est : procédures d'un programme d'analyse de propositions allemandes à exécuter après avoir trouvé la racine du verbe /geh-/ dans une proposition.

a) Le programme général :

Le programme d'analyse, dont nous allons définir ici une partie, fait partie d'un programme de traduction automatique de l'allemand en d'autres langues choisies à volonté. Nous prendrons pour base une forme du programme idéalisée, qui sans considération de la capacité limitée de l'ordinateur explique toutes les différences des constructions allemandes entre elles. De cette manière on doit regarder les propositions :

Seine Worte + gehen + in dieselbe Richtung

Herr Müller + geht + in dieselbe Richtung

comme constructions tout à fait différentes, parce que le choix du sujet leur donne une signification différente. En appliquant ce programme d'analyse au but d'une traduction automatique en une langue concrète, on peut renoncer à marquer les différences dans tous les cas où les constructions de la "langue-but" sont parallèles aux constructions de l'allemand, par exemple :

die Ware + geht + gut

tovar' + idet + xoroŝo

mais : the article + sells + well
la marchandise + se vend + bien

Le programme d'analyse consiste en plusieurs groupes de procédures, et chaque groupe est appliqué au texte entier, c'est-à-dire qu'on ne traduit pas proposition par proposition. Les groupes de procédures sont :

(1) reconnaissance des mots isolés
décomposition en lexèmes et foncteurs grammaticaux
chiffrage des lexèmes et fonsteurs par des nombres.

Le code des lexèmes consiste en un chiffre pour la classe du mot, et quelques autres chiffres, qui marquent le caractère syntactique et sémantique du mot.

(2) Interprétation des foncteurs comme catégories grammaticales (cas, nombre, temps, etc.), autant que le permet la pure morphologie.
(3) reconnaissance de la relation des éléments pronominaux aux phrases nominales précédentes.
(4) reconnaissance des adverbes presque toujours indépendants du verbe.
(cette catégorie comprend aussi adverbes locaux et temporaux, voir (5))
(5) Considération des verbes finis : le verbe fini, peut il être complété par un adverbe local ou temporal.? Par exemple :

Ich warte eine Stunde

Die Arbeit dauert einen Tag

Ich lebe in London

Dans ce cas : prendre en considération les adverbes trouvés sous (4).

(6) définition de la construction prédicative de la proposition à l'aide d'une liste de constructions prédicatives de l'allemand, établie selon les racines verbales.

4) La partie du programme communiquée ici :

Ma communication concerne la 6ème partie du programme pour le cas où la racine du verbe /geh-/ se trouve être part du verbe fini. Cette partie du programme consiste en les opérations suivantes :

(1) Recherche du préverbe devant la racine verbale ou à la fin de la proposiñon. Le préverbe sera chiffré par un chiffre général du préverbe et un nombre courant pour le préverbe spécial.
(2) Recherche des éléments ressemblants aux préverbes, mais spéciaux

(adjectifs)	(autres)
Konform	entzwei
Kaputt	irre
wund	fehl
schwanger	hoppe
fremd	zuende
müssig	vonstatten
verloren	auseinander
verlustig	fürbass
....	....
	auf und ab
	auf und davon
	auf und her
	....

L'élément préverbal sera chiffré par un chiffre général de l'élément préverbal et un nombre courant pour l' élément préverbal spécial.

(3) (Partie commune pour
propositions à /geh-/
/prév//geh-/
/el.prév//geh-/
et aussi pour d'autres verbes) .

Définition des phrases nominales et pronominales selon leurs foncteurs grammaticaux. On y trouvera des phrases nominales à tous les quatre cas purs, et d'autres avec les prépositions suivantes : als, an, auf, aus, bei, bis, bis auf, bis an, ..., durch, _____entlang, gegen, gen, in, mit, nach, über, um, unter, von, vor, wider, zu.

Cette définition selon les foncteurs grammaticaux ne suffit pas pour traduire les constructions par des constructions correspondantes Comparer par exemple les phrases nominales suivantes ;

NP + K_3	Er geht seiner Wege
	Er geht festen Schrittes
	Er geht des Geldes verlusting
nach + NP	Er geht nach Paris
	Das geht nach Gunst
	Das verb geht nach der Regel
auf + NP	Er geht auf den Bau
	Er geht auf (den) Zehenspitzen
auf + nombre	Er geht auf 80
auf + adjectif	Er geht auf arm
an + NP	Er geht an die Elbe
	Er geht mir an die Hand
	Er geht mir an den Kragen
	Er geht am Stock
	Er geht an die Milch

Dans cette partie de programme les phrases nominales seront

chiffrées par un nombre composé de chiffre général pour phrases nominales, chiffres spéciaux pour les foncteurs grammaticaux, et d'autres chiffres spéciaux pour les traits caractéristiques syntactiques et sémantiques des substantifs ou pronoms contenus dans les phrases nominales.

Après l'exécution de la 3ème partie du programme,les propositions sont rendues comme séquences de nombres complexes qui représentent phrases nominales différentes, préverbes et racine verbale. Dans le programme d'analyse il y a une liste de toutes les séquences possibles de cette manière.

Cette liste est arrangée, en succession hiérarchique, selon les caractéristiques suivantes : racine verbale; préverbe; foncteurs des phrases nominales, traits caractéristiques syntactiques et sémantiques des substantifs et pronoms. Chaque séquence sera chiffrée par

- un nombre courant, qui choisira la construction correspondante de la"langue-but" , dans une liste semblable.

- chiffres pour déterminer la fonction des phrases nominales dans la construction (paramètres).

14

SOME LINGUISTIC ARGUMENTS IN FAVOUR OF Mr BAR-HILLEL'S "NON UNIQUENESS" HYPOTHESIS

By Prof. REVZIN
Institut de Philologie Slave de l'Académie des Sciences
30 a avenue Troubnikovskï
MOSCOU

RESUME

1. In his "Four Lectures" (1963) Mr Bar-Hillel advanced a hypothesis of "different grammars for the same language lying peacefully side by side somewhere in our brain". In recent years the writer -quite independantly from Bar-Hillel (whose "lectures" were unknown to him)- came across many arguments in favour of this "non-uniqueness" hypothesis. Stylistic considerations involved were discussed in a paper read at the Semiotic Conference in Kazimierz (Poland) in September 1966. In the following some grammatical questions are briefly sketched (the whole being presented in the writers book "Methods of Modelling and Typology of Slavic Languages" - to appear in summer 1967);;

2. There are no linguistic or logic obstacles for producing words - out of morphs or simple syntactic groups (phrases) - out of words or morphs by a FS (finite state) grammar. Mr Vauquois and his colleagues have already demonstrated this fact for words in their brilliant study on the use of models in mechanical translation.

3. As for simple syntactic groups, one can get some insights into their structure by interpreting states on the diagramm of the FS grammar as linguistic categories (following in this a proposal by Bar-Hillel and Shamir from a work of 1960). For this the writer considers a restricted FS grammar (the diagramm of such grammar does not contain two arrows labelled by the same non-empty word). Under this restriction the distribution of non-eliminable empty words provides for an adequate segmenting of the group. It is

worth mentioning that there is only one restricted FS grammar for a given language (if abstraction is made from arrows labelled with the empty word).

4. The described restriction must be abandoned, if homonymous constructions are investigated. Nevertheless it will be shown that even here the distribution of the empty word labelling some arrows provides for an explanation of homonymity (e.f. there can be two different paths S_o they S_1 are $S_2 \# S_4$ flying S_5 planes S_o and S_o they S_1 are S_3 flying $S_6 \# S_5$ planes S_o, giving the sentence they are flying planes.

5. It can be assumed that the units of the level of words and that of syntactic groups (in some cases also of simple sentences) are generated not only by a device equivalent to a phrase structure grammar (e.f. a dependency grammar), but may be generated by a simplifier device (a restricted or non-restricted FS grammar), the only linguistically relevant difference being the fact that the former provides for a hierarchization of the parts (which can be semantically interpreted for all endocentric construction as a property-attributing relation), and the latter does not.

6. Almost the same applies to the relation of phrase structure grammars to transformational grammars, the latter providing for a still higher degree of hierarchization (and semantic depth).

7. Following Bar-Hillel we assume that the speaker (or the hearer) constantly switches over from one to another way of production according to the needed (or possible) degree of hierarchization. In one particular case - that of the semi-idiomatic expressions - such a solution seems obvious (one can store them in the vocabulary as whole or construct them from parts). But the most of human speech seems to be semi-idiomatic in a broader sense.

8. The non-uniqueness of syntactic description can be compared with the non-uniqueness of phonemic solutions as described first by Yuen Ren Chao and shown in its full importance in a remarkable paper by Vjach.

9. On contemplating the picture of Grammatic Activity as a constant switching over from one device to another, one is struck by the analogy to the picture of many computing devices working in parallel, which according to von Neumann ("Computers and Brain") explanesthe miracle of human intuition in genral.

15

CAN OVERT SYNTACTIC STRUCTURE BE INTERPRETED AS A LOGICAL DEVICE ?

Helmut RICHTER and Georg HINCHA
University of Cologne, Germany

RESUME

Attempts to reduce syntactic structures to logical ones, as known to the authors, tend to discover a special kind of underlying structure beyond the overt syntactic constructions. In an alternative approach, the authors try to interpret as much as possible of overt syntactic structure, as immediately representing the elements of a particular logical calculus. The paper to be presented is planned to be an extension of the authors' Syllogon Model which will be published for the first time at the Conference of the Societas Linguistica Europea in Brussels, 2nd April, 1967.

It is asked whether the syntactic elements of natural languages can be said to correspond to the elements of the calculus of multiple relations, i.e., may be identified with the particular variables, constants, and operators of that calculus. This seems to be possible not only in the general sense that relations remaining under a certain threshold of logical complexity can be verbalized. (Such verbal paraphrasing may be exemplified by REICHEMBACH classifying the preposition between as a three-place relation, obviously relying on common understanding or alleged semantics. This interpretation reminds of illustrative examples in textbooks of symbolic logic.) It should, however, also be possible to show that, regardless of content, syntactic elements organize into specific sets according to their functions as logical entities.

The authors tentatively start from the axiomatic premises that the slot of the relation variables is filled by finite verb forms and combinations of the type copulative plus nominal, whereas the slot of object variables is filled by nominals (at least those being grammatical subjects and objects). A syllogon is defined as a syntactic construction when interpreted as a predicative function. A syllogon is called a n-syllogon if it is identifiable with a n-place relation. The class of the n-sylloga is the n-syllogeme.

All syntactic structures known to be universal are either 1-sylloga or 2-sylloga. (Possibly, syllogemes of higher order may be necessary to account for constructions with subordinate sentences). In order to become a statement, generally a predicative function must be either quantified, or its variables must be substituted by individual names. While the constituents of the predicative function must be overtly present in natural syntax, the explicit quantification of a syllogon is optional. Thus, many sentences, taken in isolation, remain mere predicative functions, which imposes serious restrictions upon any attempt to detect a peculiar covert structure (of separate sentences) aimed at by some linguists under the labels of depth structure, innere Form, nomostructure, and the like. Rather, something-like "quantification by context of situation" must be conceived of.

On the other hand, many of those linguistic elements denoting a qualification are loaded with additional meaning. It is distinguished, with regard to predicative functions in natural languages, between the strictly extentional aspect of quantification and additional features in the production of statements. (We therefore subsume both quantification and additional features under the more general label of amplification.)

Variables in the different slots may be augmented by subordinate syntactic elements. It is this device which provides natural languages with the possibility to hint, within its restricted algorithm, at fairly complex relationships. Present investigation is devoted to the question how subordinate sentences can be accounted for along the lines of the present scope.

It is to be hoped that the syllogon model, with its emphasis on the overt structure as a restricted but clearly determined logics mechanism, will offer a new way of tackling with the problems of ambiguity in translation.

16

METHODS FOR OBTAINING CORRESPONDING PHRASE STRUCTURE AND DEPENDENCY GRAMMARS

Jane J. Robinson

International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT Two methods are given for converting grammars belonging to different systems. One converts a simple (context-free) phrase structure grammar (SPG) into a corresponding dependency grammar (DG); the other converts a DG into a corresponding SPG. The structures assigned to a string by a source grammar will correspond systematically, though asymmetrically, to those assigned by the target grammar resulting from its conversion. Since both systems are weakly equivalent, generating exactly the CF languages, the methods facilitate experimentation with either notation in devising rules for any CF language or any CF set of strings designed to undergo subsequent transformation.

A source SPG is assumed to be of finite degree with ordered rules in which only the initial symbol is recursive. Unless the source grammars obey additional constraints, the target grammars may exhibit a peculiar property, defined as "structure sensitivity". The linguistic implications of the property are discussed, and the linguistic motivation for imposing the constraints necessary to avoid its appearance is suggested.

The author owes an especial debt to Jesse Wright of the Automata Theory and Computability Group in the Mathematical Science Department of IBM Research for many helpful discussions of theoretical problems arising in the course of this investigation.

In an article on dependency theory[1] appearing in 1964 [5], Hays remarked, "Casual examination suggests there would be little difference between transformation of dependency trees and transformation of IC [immediate constituent] structures, but no definite investigation has been undertaken." Since then, a dependency grammar with transformational rules has been designed for a subset of English sentences, and preliminary results indicate that Hays' observation is correct. [9] In either case, transformations are specified in terms of labeled trees, and the number of branches and the denotations of the labels do not affect the essential operation. Since the base grammar is generally limited to the generation of context-free pre-terminal strings of "deep structure", and since context-free languages are characterizable in either dependency or phrase structure (i.e., immediate constituent) systems, neither system is clearly preferable as a base. A linguist may find that the notation afforded by one or the other is simpler for characterizing some language, or for defining structures to be transformed, or for adapting a grammar to computer applications. A linguist may also wish to experiment with grammars of both types, or redesign transformations defined on the structures of one base grammar in order to incorporate them into a transformational grammar using a different base.

Such considerations as these motivate the present treatment of the problem of obtaining paired grammars by converting a grammar of one kind into a systematically corresponding grammar of the other which generates the same sentences and assigns comparable structures. In addition, conversion draws attention to some linguistically significant relationships that may exist unnoticed among the categories and rules of the source grammar and which may induce in the derived grammar a peculiar property of *structure sensitivity*, roughly analogous to context sensitivity. This property will be exhibited and discussed in the course of illustrating the method.

We begin concretely by inspecting (Fig. 1) a pair of grammars: EPG1, a simple or context-free phrase structure grammar of the kind formalized by Chomsky [2], Bar-Hillel [1], and others, and EDG1, a dependency grammar of the kind formalized by Gaifman [4] and Hays [5]. Two structural diagrams, a P-tree and a D-tree drawn beneath the grammars, illustrate the structure each assigns.

The rewriting rules of EPG1 are of two kinds, those in which only categories appear, and those in which a category is rewritten as a terminal (lexical formative or word). The latter may be separated from the former and made into assignment rules,

---

[2] For additional material on dependency theory, see Ref. 6.

SPG1

Axiom: # S #

Rewriting Rules:

1. S → NP VP
2. VP → V NP
3. NP → D N
4. D → the
5. D → some
6. N → boys
7. N → girls
8. V → like
9. V → admire

(1a)

DG1

Axiom: * (V)

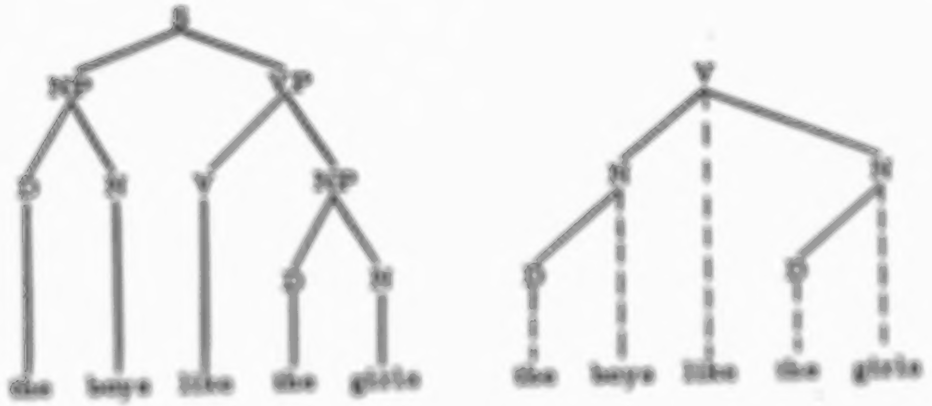
Dependency Rules:

1. V (N * N)
2. N (D *)
3. D (*)

Assignment Rules:

1. D: the, some
2. N: boys, girls
3. V: like, admire

(1b)


S
NP
VP
D
N
V
NP
D
N
the
boys
like
the
girls
V
N
N
D
D
the
boys
like
the
girls

(1c) (1d)

Figure 1

thereby increasing the resemblance of SPG1 to DG1 in an obvious way. This is possible because SPG1 does not contain any mixed rewriting rules in which both categories and lexical formatives appear on the right. It can be shown that any SPG which has mixed rules, in this sense, can be converted into one which does not merely by introducing new categories, without affecting generative power. Thus there is no reason for not separating the two types of rules, and Chomsky devotes a good part of Aspects of the Theory of Syntax [3] to giving linguistic reasons for just such a separation.

Hereafter, "rewriting rules" will refer only to those like rules 1-3 in SPG1, and those like rules 4-9 will be called "assignment rules". Categories which do not appear on the left of any rewriting rules are terminal categories.

With each category of SPG1, we associate a number called its degree. (To say that a category is of degree n means that n is the fixed upper limit to the number of nodes of the shortest path leading from it to a terminal category in any structure derived from it by successive rule applications. A category may be of infinite degree. For example, if X → XX, then X is of infinite degree and so is the grammar in which it occurs even though another rule rewrites X as a string containing only terminal categories.)[1] In SPG1 the terminal categories D, N, V are of zero degree, since they are not rewritten; the categories NP and VP are of degree 1, since at least one category in their replacement is of zero degree; and the degree of S is 2, since the least degree assigned to any category on the right of the rule for rewriting S is 1. The maximum number assigned to any category in a grammar is also the degree of the grammar; therefore, the degree of SPG1 is 2.

An essential difference between SPG1 and DG1 now emerges more clearly. DG1 uses only terminal categories, while SPG1 uses categories of higher degree. The effects of the differences are reflected in the structure of the P-tree and D-tree. The latter has fewer branches, and this will always be the case for structures assigned to the same string by grammars belonging to the two different systems.

Even so, there is a systematic correspondence between the two trees and their labels of a kind defined by Hays [5]. Every complete subtree of the D-tree, that is, every node taken together with all of its descendents, covers a substring of the sentence that is covered by a complete subtree of the P-tree. The converse does not hold, but every complete subtree of the P-tree covers a substring that is covered by a connected subtree of the D-tree. In the example, both complete subtrees dominated by N in the D-tree correspond to the two complete subtrees of the P-tree dominated by NP, and the complete (sub)tree dominated by V corresponds to the complete (sub)tree dominated by S. However, the complete subtree dominated by VP in the P-tree corresponds to an incomplete subtree of the D-tree, which is dominated by V but includes only V and the branches to its right; so that the relationship of correspondence is asymmetrical.

---

[1] For a more precise characterization of "degree", see Gaifman [4].

While such systematic correspondences exist between the structures assigned to all strings generated by SPG1 and DG1, this is not the general case. In general:[1]

1. Any context-free language can be generated by grammars of either simple phrase structure or dependency systems.
2. For any given SPG, there exists one or more DG over the same language all of whose structures correspond systematically to structures assigned to the same strings by the SPG if and only if the SPG is of finite degree.
3. For any given DG there exists one or more corresponding SPG.
4. For any given DG there exists a unique SPG of degree 1 that is strongly equivalent to it.

Gaifman [4] gives a very general method for constructing corresponding DG from any SPG of finite degree, and also a method for constructing a unique SPG of degree 1 from any DG. Here we give two methods for constructing corresponding SPG and DG that differ from Gaifman's. The first applies only to a more restricted set of SPG and leads to reduced DG, whereas Gaifman's method tends to produce DG with overlapping categories and superfluous rules that may never be used to generate any string. The second allows construction of SPG of degree greater than 1 from certain DG.[2]

A simplified sketch of the two methods follows. Each rewriting rule is "augmented" by starring a category on the right. Each dependency rule is augmented by assigning a numerical coefficient to each dependent. Figure 2 shows a possible augmented form of SPG1 on the left and a possible augmented form of DG1 on the right. (Detailed consideration of the problem of augmentation will follow the sketch of the operations.) These augmentations were deliberately chosen so that conversion of either grammar uniquely produces the other. Different augmentations produce different results, although structures of the original grammar and of the grammar derived from it still correspond.

---

[1] Proofs are given by Gaifman [4]. It is assumed that the SPG is non-erasing and reduced; that is, no category is rewritten as null, and there are no superfluous categories or rules.

[2] The second method was suggested by Hay's procedure for constructing P-trees from D-trees [7]. More precisely, an SPG of degree $n > 1$ may be derived if, in the DG, some categories govern two or more dependents, and the left- or rightmost dependent itself governs dependents.

SPG1	TS	IRL	DG1
S → NP VP*	$S{:}V^2$	$V^2 \rightarrow N^1 \ V^1_*$	V (2N * 1N)
VP → V* NP	$VP{:}V^1$	$V^1 \rightarrow V^0_* \ N^1$	N (1D *)
NP → D N*	$NP{:}N^1$	$N^1 \rightarrow D^0 \ N^0_*$	D (*)

Figure 2

The columns TS and IRL in Figure 2 represent a Table of Substitutes and an Intermediate Rule List. In the conversion of SPG1, the TS is constructed, equating each category on the left of a rule with a superscripted terminal category. The numerical superscript (hereafter called the exponent) equals the number of rules traced through when tracing by starred categories before a starred terminal category is reached, and expresses the distance between the terminal category and the category for which it is a substitute. In the IRL, the categories occurring in each rule of the SPG are replaced by their substitutes from TS. Taking the first IRL rule, construction of a dependency rule is begun by replacing the arrow with parentheses enclosing the substitute categories on the right. Thus from the first IRL rule in Figure 2 we obtain V^2 (N^1 V^1_*). The construction of a dependency rule is continued so long as the exponent of the starred category X^n_* is greater than zero. The next step is to search the IRL for a rule with X^n on the left. The categories on the right of the new rule are inserted in the parentheses of the D rule under construction, in the position of X^n_*, but no new parentheses are added. When the starred occurrence is X^0_*, it is replaced by *, all exponents are erased, and the dependency rule is complete. The process is repeated for new dependency rules until all IRL rules are exhausted.

We must add rules to the constructed DG for any unstarred terminal category of the IRL. The added rules assign no dependents and are of the form X(*). The assignment rules are simply transferred, and since V is the substitute for the axiom category S, it is taken as the axiom for the DG.

Going in the other direction from an augmented DG1 to

SPG1, we first assign an exponent s to each category, where s equals the largest coefficient of any dependent of that category. If a category is not assigned dependents, its exponent is zero. The first rule of augmented DG1 now appears as $V^2\ (2N^1\ *\ 1N^1)$. From this rule two rules are constructed for the IRL; that is, the number of IRL rules constructed from each rule of DG1 will be equal to the exponent of the governor. The first rule so obtained writes the governor with its exponent on the left of the arrow. All of its dependents whose coefficients equal that of the exponent are written in order on the right, and the governing category with its exponent decreased by 1 is written with the *. Thus we obtain $V^2 \rightarrow N^1\ V^1_*$. The second rule is obtained in a similar fashion, with the exponent of the governing category diminished by 1 yielding $V^1 \rightarrow V^0_*\ N^1$. The process is repeated until an IRL rule rewriting some category with exponent equal to 1 is used, after which the next DG rule is processed, and so on until all DG rules are exhausted.

At this point, the *'s may be erased and, except for category labels, the IRL is exactly equivalent to the rewriting rules of the original unaugmented SPG. The only function the TS serves is to reassign labels. Assignment rules are transferred and the substitute for the axiom of DG1 is added.[1]

SPG1 and DG1 are very simple, with no embeddings and no optional rules. More complicated grammars give rise to problems of augmentation, especially for SPG. Even SPG1 poses a problem. Assume it had been augmented by starring NP in rule 1 and in rule 2. In that case, the same substitute, N^2, is assigned to both S and VP, and the procedure produces a DG that is not even weakly equivalent to the original. Clearly some constraints must be imposed on augmentation and provision made for grammars in which it may not be possible to avoid starring a category more than once.

Similarly, assume that some DG has a rule of the form C (A B * D), and that this is augmented as C (1A 2B * 1D). The IRL rules are:

$$C^2 \rightarrow B\ C^1_*$$
$$C^1 \rightarrow A\ C^0_*\ D.$$

But now the rules generate the sequence B A C D, which was not generated by the original rule, and do not generate A B C D, which was. This is remedied by requiring that if a coefficient n has been assigned to a dependent, no higher number is assigned to any dependent which intervenes between it and the *. We will also require that at least one dependent be preceded by 1, and if any dependent is preceded by $n > 1$, there must be at least one dependent preceded by n - 1. This is not crucial, but it avoids setting up unnecessary, single-branching categories in the derived SPG. Note that if all dependents in any rule are preceded by 1, which is the same as not augmenting the DG at all, the resulting SPG will be of degree 1; that is, each rule of the grammar will contain at least one terminal category on the right. This is equivalent to Gaifman's procedure [4].

---

[1] Although grammars with only one axiom are illustrated, grammars with more than one axiom can obviously be handled as well.

Augmentation of SPG is the more difficult case. Primarily it is the problem of constructing the TS in a finite number of steps. For example, if for S → NP VP S the S is starred on the right, an infinite loop is created immediately. This is easy to avoid when considering a single rule or a small set of rules, but we do not know in general whether some series of rule augmentations may not lead to the same situation. Gaifman's solution is to require that the marked category be of lesser degree than the category on the left, but this not only leads to a proliferation of categories in the derived DG whenever the SPG has more than one rewriting rule for any of its categories, it prevents us from deriving the simplest corresponding DG in some cases.[1] On the other hand, the method employed here will not work unless some restrictions are imposed on augmentation which also imply restrictions on the form of the SPG in addition to the requirement of finite degree. It is not clear what restrictions are minimally necessary, but it is sufficient to require, in addition to finiteness of degree, that the rules of the SPG be ordered, so that in a developing derivation, if rule n has been applied, no rule m, $m < n$, need be applied thereafter. This is too restrictive, and does not allow for full recursion. We may, however, allow a dummy symbol, S', to appear in any rule rewriting some X as a string containing some Y, $S' \neq Y \neq X$, where S' is replaced after one application of the set of rules by # S #, the axiom, so that the rules then reapply in linear order.

Any SPG of finite degree with ordered rules providing for recursive application in the manner specified can be converted to a corresponding DG by the method given here. The base component proposed by Chomsky [3] for transformational grammar is an SPG of this form.[2]

---

[1] "Simplest" with respect to process, number of categories, and freedom from the property of structure sensitivity. See p. 14 ff.

[2] Chomsky does not explicitly require that if S' appears in rewriting X, some Y ($S' \neq Y \neq X$) appear in the rule also, but implicitly observes the restriction. A new grammar for English by Rosenbaum [10] contains a rule NP → NP S' which does not observe it. This is the only case known to me of an actual grammar for some "real" language that violates this restriction, and Rosenbaum does not claim any strong theoretical motivation for the rule in question. Cf. also Lees [8], and the MITRE grammar [11].

We turn now to more detailed consideration of augmenting and converting SPG subject to the above constraints. While the constraints insure that the SPG can be workably converted, some linguistically significant problems arise if we consider how to derive a simple DG, and it will be shown that under some conditions the derived DG will exhibit a feature not heretofore considered in the literature, a feature of "structure sensitivity".

Method 1. Conversion of SPG to DG

Step 1. Augmentation

All rules of the SPG which rewrite a given category are conflated and written as one schematised rule, with square brackets enclosing optionally omitted categories and braces enclosing lists of categories from which a single choice is made.[1] Thus $Z \rightarrow [W] \begin{Bmatrix} X \\ Y \end{Bmatrix}$ conflates four rules, and will, in any given application, rewrite Z as either WX or WY or X or Y.

Beginning with the first rule rewriting some X and proceeding in rule order, star occurrences of categories, excluding the dummy symbol S', on the right in such a way that one and only one starred category Y*, where Y ≠ X, will occur in any application of the rule.[2] It follows that no bracketed (omissible) occurrence may be starred. If more than one category is starred, the schematized rule must be separated into as many rules as there are starred categories, with one starred category appearing in each. For example,

$X \rightarrow \begin{Bmatrix} A^* & \begin{Bmatrix} B \\ C \end{Bmatrix} \\ D^* & F \end{Bmatrix}$ must be separated into $X \rightarrow A^* \begin{Bmatrix} B \\ C \end{Bmatrix}$ and $X \rightarrow D^*\ F$. This is a partial undoing of the conflation, but for linguistic reasons all ways of rewriting a category will usually have some element in common and some conflation will usually be preserved.

---

[1] Square brackets are used here rather than the customary parentheses to avoid confusion with the parentheses used in the derived DG. More exactly, [ ] and { } enclose lists of strings of categories. In the case of [ ] the list may consist of one member, and in both cases the strings may consist of one item.

[2] Since the SPG must be finite, no rule will rewrite X as a string of X's in any application, and some other category of degree less than X will be available for starring in all cases.

If the simplest DG corresponding to the SPG is desired, then it is preferable to augment the SPG in such a way that

a) no category occurs both starred and non-starred on the right, and
b) no category is starred on the right of two or more rules which rewrite different categories.

But it is not always possible to observe these policies, and in that case, additional augmentation is necessary in order to distinguish among occurrences of X as the marked constituent in the rule rewriting Y, X as the marked constituent in the rule rewriting Z, $Z \neq Y$, and X as an unmarked constituent of any category.

Assume some category X is starred in two or more rules which rewrite different categories,[1] say Y and Z, and also occurs unstarred on the right in some rules. The two X*'s are assigned different subscripts to avoid assigning the same substitutes for Y and Z with the consequent loss of essential information. We now have three varieties of X, namely: X, X_1, and X_2, where X is the unstarred variety, X_1 is the marked constituent of Y, and X_2 of Z. (If X does not occur unstarred, only X and X_1 are needed.) If X is not a terminal category, there is a rule rewriting X. Then we must add, beneath that rule, rules for rewriting X_1 and X_2. If $X \rightarrow U^* \ W$, we add $X_1 \rightarrow U_1^* \ W$
and $X_2 \rightarrow U_2^* \ W$.
Now, if U is not a terminal category, we add rules for rewriting U_1 and U_2 beneath the rule rewriting U. Thus the process of adding rules is iterative, but it will eventually end when, in some lower rule, a terminal category is starred.

Note that in some cases sub-subscripts are needed. For example:

1. $Z \rightarrow \ldots P_1^* \ldots$
2. $X \rightarrow \ldots R_1^* \ldots$
3. $Y \rightarrow \ldots R_2^* \ldots$
4. $R \rightarrow \ldots P_2^* \ldots$
5. $P \rightarrow \ldots A^* \ldots$

---

[1] X* may occur more than once in rules rewriting the same category, Y.
E.g., in $Y \rightarrow \left\{\begin{matrix} X & Y \\ Y & X \\ Z & \end{matrix}\right\}$, the only possible starring is
$Y \rightarrow \left\{\begin{matrix} X^* & Y \\ Y & X^* \\ Z^* & \end{matrix}\right\}$, which produces three rules for Y, in two of which X* occurs.

Proceeding down the rules, we see first that P_1* occurs, and scanning down the left, that P's are not terminal categories and that there is no rule rewriting P_1. Therefore,

beneath $P \rightarrow \ldots A* \ldots$

we add $P_1 \rightarrow \ldots A_1* \ldots$.

In the second rule, R_1* occurs, is non-terminal, and requires a rule, so

beneath $R \rightarrow \ldots P_2* \ldots$

we add $R_1 \rightarrow \ldots P_{2_1}* \ldots$.

R_2* occurs in the third rule, under the same conditions, and

beneath $R_1 \rightarrow \ldots P_{2_1}* \ldots$

we add $R_2 \rightarrow \ldots P_{2_2}* \ldots$.

P_2* occurs in the fourth rule, under the same conditions, and

beneath $P_1 \rightarrow \ldots A_1* \ldots$

we add $P_2 \rightarrow \ldots A_2* \ldots$

Our rules are now:

1. $Z \rightarrow \ldots P_1* \ldots$
2. $X \rightarrow \ldots R_1* \ldots$
3. $Y \rightarrow \ldots R_2* \ldots$
4. $R \rightarrow \ldots P_2* \ldots$
5. $R_1 \rightarrow \ldots P_{2_1}* \ldots$
6. $R_2 \rightarrow \ldots P_{2_2}* \ldots$
7. $P \rightarrow \ldots A* \ldots$
8. $P_1 \rightarrow \ldots A_1* \ldots$
9. $P_2 \rightarrow \ldots A_2* \ldots$.

and we are looking at rule 5 where $P_{2_1}*$ requires us to add

10. $P_{2_1} \rightarrow \ldots A_{2_1}* \ldots$.

Rule 6 requires us to add

11. $P_{2_2} \rightarrow \ldots A_{2_2}* \ldots$.

But in rule 7, A* requires no additions, because A is not subscripted, and the subscripted A's in rules 8-11 require no additions because A's are terminal categories.

As a result, no X_i* occurs on the right of two or more rules which rewrite different categories.

This process and the remaining steps will be illustrated using EPG4. In order to show the effects of different choices in augmentation, SPG2 will be augmented in a way that deliberately violates the policies advocated for starring (but not the restrictions).

SPG2

Axiom: S

Rewriting Rules:

1. $S \rightarrow NP^* \ VP$
2. $VP \rightarrow V \ NP^* \ [NP]$
3. $NP \rightarrow D \ N^*$

We will assume assignments are the same as for SPG1 with the additional assignments of "send" and "give" to V, and "books" and "flowers" to N.[1]

Note that, in this augmentation, NP is starred twice and also occurs non-starred on the right. Subscripting and duplication are required, resulting in

1. $S \rightarrow NP^*_1 \ VP$
2. $VP \rightarrow V \ NP^*_2 \ [NP]$
3. $NP \rightarrow D \ N^*$
4. $NP_1 \rightarrow D \ N^*_1$
5. $NP_2 \rightarrow D \ N^*_2$

Step 2. Establish a table of substitutes (TS) of 2 columns and n rows, where n equals the number of rules in the SPG (after step 1). List categories on the left in column 1, and starred categories on the right in column 2, in order. Eliminate any duplicate rows. (Cf. p. 9, footnote 1.)

At the end of step 2, applied to SPG2, the result is

TS

S	NP_1
VP	NP_2
NP	N
NP_1	N_1
NP_2	N_2

---

[1] In a transformational grammar, lexical assignment rules of more complex form than that given here would presumably block the generation of un-English sentences.

Step 3. Starting with k = 1, try to match the category in row k, column 2 (k, 2) with a category occurring in column 1 of a lower row. If a match is found on row m, check to see if a match also occurs on m + 1. (This will be the case if the SPG contains more than one rewriting rule for the category in (k, 2).) If it does, mark m as a branching point, insert a duplicate of row k beneath row k (the duplicate will be row k + 1) and follow separate branches for substitutes for (k, 1) and (k + 1, 1). Replace the category in (k, 2) with the category in (m, 2) and repeat the search on the remaining lower rows. When the search is exhausted, assign an exponent e to the last category obtained (the final substitute) in column 2, where e equals the number of matches plus 1. Increment k and repeat until every category in column 1 has a substitute that does not appear in column 1. At the end of step 3, we obtain a unique substitute for every rewritten category of SPG3.[2]

TS

S	N_1^2
VP	N_2^2
NP	N^3
NP_1	N_1^3
NP_2	N_2^3

Step 4. Construct an Intermediate Rule List (IRL) by replacing each category in the augmented SPG rules with its substitutes from TS. If a category has no substitute, superscript it with a zero. If X_i has more than one substitute, include them in braces wherever X_i appears on the right. If a category X occurring on the left has more than one substitute, provide a separate rewriting rule for each substitute Y_k^e such that $Y_k^e \rightarrow \ldots Y_k^{e-1} \ldots$.[3] If a non-starred category on the right

has more than one substitute, include all substitutes as braced options. At the end of step 4, we obtain

---

[2] In cases, not illustrated here, where several substitutes are found because several rules rewrite some X_i, each substitute will be uniquely assigned to X_i.

[3] E.g., assume $X \rightarrow \ldots AX \ldots$
$X \rightarrow \ldots BX \ldots$
$Y \rightarrow \ldots BX \ldots$
$Y \rightarrow \ldots CX \ldots$
so that the substitutes for X are A^3, B^3, and C^3, and the substitutes for Y are B^3 and C^3. Then the IRL is:

1. $N_1^2 \rightarrow N_1^1* \; N_2^2$
2. $N_2^2 \rightarrow V^0 \; N_2^1* \; [N^1]$
3. $N^1 \rightarrow D^0 \; N_*^0$
4. $N_1^1 \rightarrow D^0 \; N_1^0*$
5. $N_2^1 \rightarrow D^0 \; N_2^0*$

Step 5. Take the first unmarked IRL rule, which rewrites some X_n^s, set s as a counter, and write X_n followed by a pair of parentheses enclosing the string of categories on the right of the IRL rule. Note that the string will contain an $X_n^{s-1}*$.

Step 6. Mark the previously processed IRL rule, decrease s by 1, and test for $s = 0$. If so, go to step 7. Otherwise, find the rule which rewrites the new X_n^s. Replaced the starred X_n^s* in the current D rule with the categories on the right of the IRL rule. Repeat step 6.

Step 7. Test to see if any unmarked rules remain in the IRL. If they do, return to step 5. If not,

a. Erase starred categories, leaving only the star.
b. Add rules of the form X(*) for any non-starred terminal category of the IRL.
c. Add as axiom(s) of the DG the substitute(s) of the axiom(s) of the SPG.
d. Add the assignment rules of the SPG. If a category is subscripted in the DG, the assignments are duplicated for each subscripted variant.
e. Erase exponents.

At the end of step 7, we obtain the derived DG2.

Axiom: $* \, (N_1)$

Dependency Rules:
1. $N_1 \; (D \; * N_2)$

---

2. Then the IRL is:

$A^1 \rightarrow \ldots A\overset{0}{*} \ldots$
$B^2 \rightarrow \ldots B\overset{1}{*} \ldots$
$C^2 \rightarrow \ldots C\overset{1}{*} \ldots$
$B^1 \rightarrow \ldots B\overset{0}{*} \ldots$
$C^1 \rightarrow \ldots C\overset{0}{*} \ldots$

2. N_2 (V D * [N])

3. N (D *)

4. V (*)

5. D (*)

If we interpret each distinct X_n as a separate category, then the same list of words is assigned to the three categories, N, N_1, and N_2, by the assignment rules. If we interpret them as the same category, then the subscripts distinguish the different substructures of a sentence in which the N's occur. Each N governs a D directly on the left, but if it is the axiom it is required to govern another N on the right also. If N is not the axiom but is governed directly by the axiom, it may govern another N on the right, and is required to govern a V on the left. If it is neither the axiom nor a direct dependent of the axiom, it governs nothing but the D. We may not erase the subscripts and write a single conflated rule N ([V] D * [N]), for then strings not generated by SPG2 would be generated. (For example, the rule would generate an infinite set of strings, $(D\ N)^n$.) In such circumstances we say that the DG is structure sensitive.

Definition 1. A DG is structure sensitive if

a. the set of terminals assigned to one category is identical to the set assigned to any other category, and/or

b. any rule restricts the choice of dependents a category may govern to a subset of the ordered dependents it is permitted to govern in some other rule.

Note that a DG containing the rules A (*) and A (B * C) is structure sensitive by this definition. Here, too, conflation is impossible, since A ([B] * [C]) allows A to govern B without governing C.

This structure-sensitive feature of some DG apparently serves functions like those served by the context-sensitive feature of context-sensitive phrase structure grammars in placing restrictions on the string generated, but there seems to be no mention of it in the literature. Its character may be masked by the freedom to set up categories and assign the same terminals to them.[1] For example, one may substitute simple symbols X and Y for the complex symbols N_1 and N_2, and obtain the rules

X (D * Y)
Y (V D * [N])
N (D *)

---

[1] Gaifman's method [4] for converting SPG to DG makes great use of this freedom.

In this case, only the assignment rules, assigning exactly the same set of terminals to X, Y, and N, explicitly show the structure sensitivity, although the fact that N governs a subset of the dependents of X and Y is significant.

Such arbitrariness in assigning symbols raises the significant linguistic problem of criteria for establishing categories, which is too large an issue to be discussed here. The problem is no less relevant to SPG.

Definition 2. An SPG is structure sensitive if
a. the set of terminals assigned to one terminal category is identical to the set assigned to any other, and/or
b. any rewriting rule does not contain, on the right, a unique category (other than the axiom) which occurs only once in that rule and appears on the right in no other rule.

The linguistic implications of the property may be clarified by considering two sets of rules, one for the artificial language $a^n b a^n$ and one for a fragment of English.

The language $a^n b a^n$ is generated by the rules $S \rightarrow A\ S\ A$, $S \rightarrow B$, $A \rightarrow a$, and $B \rightarrow b$. The first rule "does not contain a unique category (other than the axiom) which occurs only once", since A occurs twice. One of the A's must be starred in converting to a DG and the DG will distinguish two categories of a, a left A and a right A. Note that the same language is generated if the first rule is $S \rightarrow A\ S$ and a transformational rule $X\ B \Rightarrow X\ B\ X$ is added. In this case, each rewriting rule contains a unique category other than the axiom, and a structure-free set of dependency rules is obtainable from them.

A less artificial but still extreme case of structure sensitivity is that in which two or more rules are rewritten in exactly the same way. Assume that an SPG for English has the following rewriting rules:

$S \rightarrow X\ VP$
$VP \rightarrow V\ Y$
$X \rightarrow D\ N$
$Y \rightarrow D\ N$

Here the distinction drawn between a sequence D N that is a(n) X (i. e., derives from X) and a D N that is a Y, reflects the functional notions of subject and object, but obscures the categorial notion is a noun phrase. Chomsky [3, pp. 68-72] argues that it is confusing and redundant to assign categorial status to both notions since the purely relational character of the functional notion is implicit in the rewriting rules $S \rightarrow NP\ VP$ and $VP \rightarrow V\ [NP]$; that is, the notion "subject of a sentence" refers to the NP under the immediate domination of S, and "object of a verb" refers to the NP under the immediate domination of VP. Chomsky also shows that for sentences like "John was persuaded by Bill to leave" where "John" is simultaneously object-of "persuade" and subject-of a transformed embedded sentence "John leave", it is impossible to represent such functional notions by categorial assignments, and adds that "Examples of this sort, of course, provide the primary motivation and empirical justification for the theory of transformational grammar." [p. 70].

Whether it is possible or desirable to require that SPG components of transformational grammars for natural languages be structure free is an open question. Presumably, it is desirable if it is possible, since the least powerful, most restricted grammar -- the tightest fit -- is to be preferred. Moreover, inspection of proposed grammars, for English at least, indicates that most of their rules do contain unique categories on the right.

Returning now to SPG2, we see that it is structure free, since every elementary rewriting rule for every category X_i contains a single occurrence of at least one category Y_i on the right which does not appear elsewhere on the right, and is not the axiom. Under these conditions we will say that Y_i is a head for category X_i and call all such Y's head categories. Examination shows that the structure-sensitive property of DG2 arose from the choices made in augmenting SPG2. If only head categories are marked, a structure-free DG similar to DG1 results, in which all signs of augmentation can be erased without altering its generative capacity.

Intuitively, it seems reasonable to regard heads as sources of a "governor" in any string derivable from the category in whose rewriting rules they appear. This does not mean that the string is to be considered endocentric in the strong sense of requiring that the governor be substitutable for the entire string without loss of grammaticality, and the objection sometimes raised that dependency theory forces a purely endocentric analysis of a language is based on failure to distinguish between "head of" and "substitutable for". It appears truer to say that dependency analysis assumes that one phrase type is distinguished from another primarily by the singular presence of some category in it rather than by co-occurrence and order of categories in it.

Incidentally, aside from the problem of obtaining a structure-free DG from an SPG, avoidance of structure sensitivity may be a criterion for assigning government when one is analyzing a language in terms of dependency theory. In English, for example, the choice has generally wavered between noun and verb as candidates for sentence government. Since every elementary sentence contains one verb but may contain several nouns, choosing the noun forces a structure-sensitive DG.

In converting a DG to an SPG, no requirement of intrinsic ordering needs to be imposed on the dependency rules, as it was on the rewriting rules of SPG. Dependency rules may always be partially ordered by starting with the rule (or rules) for the axiom(s). Call these "level zero rules". Then level 1 rules are those which assign dependents to the axiom, level 2 those which assign dependents to categories which make their first appearance in level 1 rules, and so on. To insure eventual termination, however, it is required that if X occurs anywhere in a level n rule, and is a dependent in a level m rule, $m \geq n$, then its choice as dependent in the level m rule is optional or else the governor in the level m rule is optionally chosen at some point. Otherwise no constraints on recursion are necessary, and any category may be reintroduced at a lower level.

By contrast, the problem of conflation arises. It has been shown that rules like A (*) and A (B * C), occurring in structure-sensitive DG, cannot be conflated. Conflations of A (B *) with A (C * D) and of A (* B) with A (B *) are also impossible, although structure sensitivity as defined above is not involved. If the rules are not conflatable because the DG is structure sensitive, then the SPG may also be structure sensitive. If they are not conflatable solely because of disparate number or position (left or right) with respect to the governor, the SPG will be structure free. In either case, the conversion process becomes somewhat more complicated.

The process will be illustrated first by applying it to a DG whose rules cannot be fully conflated both because some are structure sensitive and because some assign disparate numbers or positions to dependents of a category.

Method 2. Conversion of DG to SPG

Step 1. Augmentation

Definition: A dependent element in a dependency rule is any braced or bracketed string not included in larger braces or brackets, and any single unbraced, unbracketed occurrence other than * occurring within parentheses.

E.g., in $A \left(\begin{Bmatrix} B\ [C] \\ D \end{Bmatrix} * E\ [F] \right)$ there are three dependent elements: $\begin{Bmatrix} B\ [C] \\ D \end{Bmatrix}$, E, and [F].

a. Precede each dependent element with a coefficient $n \geq 1$, so that for any $n > 1$, at least one element is preceded by $n - 1$, and for any $m > n$, m does not intervene between n and *. Thus:

A (2B * 1C)
A (2B 1C *)
but not A (2B * 2C)
and not A (1B 2C *).

b. Assign an exponent s to the governor in each rule, where s equals the largest coefficient n of any dependent. (For rules of the form X (*), s is zero.)

c. If a category X is assigned different exponents for different rules in which it occurs as governor, associate a distinct subscript with each distinct exponent.

d. Replace every occurrence of X as dependent by X^s. If X has been subscripted, include each variant in braces, thus:

$$\begin{Bmatrix} X_1^{s_i} \\ X_2^{s_j} \end{Bmatrix}$$

Step 2. Test for unmarked (unprocessed) rules. If none remain, go to step 4. Otherwise take the first unmarked rule, where some $X_n^{s_i}$ is governor and set a counter $S = s_i$.

Step 3. If $S = 0$, mark the rule and repeat step 2. If $S \neq 0$, construct a rule of an Intermediate Rule List of the form $X_n^S \rightarrow \ldots$. On the right of the arrow duplicate all dependent elements whose coefficients $n = S$. Include the *, and precede it with X_n^{S-1}. Decrease S by 1, and repeat step 3.

Step 4. Establish a Table of Substitutes, assigning a unique symbol to every distinct $X_n^{S_i}$ for which $S_i \neq 0$.

Step 5. Assign the substitute or substitutes for the axioms of the DG as axioms of the SPG.

Step 6. Rewrite the IRL, using substitutes from the TS, deleting exponents and subscripts from any X_n^0, and omitting *.

Step 7. Transfer the assignment rules.

This method will be illustrated using an abstract DG3, without assignment rules. DG3 is structure sensitive since there are rules which restrict the choices of dependents of categories A and C to subsets of the ordered categories they are permitted to govern in other rules.

DG3 (augmented)

Axiom: * (A)

Dependency Rules:[1]

1. A (1B *)
2. A (2 $\left\{\begin{matrix} B \; D \\ C \end{matrix}\right\}$ 1B * 3E)
3. B (1E *)
4. B (* 1F)
5. C (1G * 1F)
6. C (*)
7. D (*)
8. E (* 1G)
9. F (1A *)
10. G (*)

The final augmented form after step 1 is:

Axiom: * $\left\{\begin{matrix} A_1^1 \\ A_2^3 \end{matrix}\right\}$

Dependency Rules:

1. A_1^1 ($1B^1$ *)
2. A_2^3 (2 $\left\{\begin{matrix} B^1 \quad D^0 \\ C_1^1 \\ C_2^0 \end{matrix}\right\}$ $1B^1$ * $3E^1$)
3. B^1 ($1E^1$ *)
4. B^1 (* $1F^1$)
5. C_1^1 ($1G^0$ * $1F^1$)
6. C_2^0 (*)
7. D^0 (*)
8. E^1 (* $1G^0$)

---

[1] Note the impossibility of further conflation, and that every category occurring as dependent in a rule of level m and also occurring in some other rule of level $n \leq m$ is always optionally chosen as a dependent. That is, there is the possibility of avoiding its reintroduction. Thus A, the axiom, is reintroduced in rule 9, but its governor, F, is optionally chosen as a dependent. Otherwise no derivation could terminate.

9. $F^1\ (1\left\{\begin{matrix}A_1^1\\ A_2^3\end{matrix}\right\}*)$

10. $G^0\ (*)$

At the end of iterations on steps 2 and 3, the IRL is

1. $A_1^1 \rightarrow B^1\ A_1^0{*}$
2. $A_2^3 \rightarrow A_2^2{*}\ E^1$
3. $A_2^2 \rightarrow \left\{\begin{matrix} B^1\ D^0 \\ C_1^1 \\ C_2^0 \end{matrix}\right\} A_2^1{*}$
4. $A_2^1 \rightarrow B^1\ A_2^0{*}$
5. $B^1 \rightarrow E^1\ B_*^0$
6. $B^1 \rightarrow B_*^0\ F^1$
7. $C_1^1 \rightarrow G^0\ C_1^0{*}\ F^1$
8. $E^1 \rightarrow E_*^0\ G^0$
9. $F^1 \rightarrow \left\{\begin{matrix} A_1^1 \\ A_2^3 \end{matrix}\right\}\ F_*^0$

A possible TS is:

$S : A_1^1$

$Z : A_2^3$

$Y : A_2^2$

$X : A_2^1$

$W : B^1$

$V : C_1^1$

$U : E^1$

$T : F^1$

SPG3 is:

Axiom(s): $\left\{\begin{matrix} S \\ Z \end{matrix}\right\}$

Rewriting Rules:

1. $S \rightarrow W\ A$
2. $Z \rightarrow Y\ U$
3. $Y \rightarrow \left\{\begin{matrix} W\ D \\ V \\ C \end{matrix}\right\} X$
4. $X \rightarrow W\ A$
5. $W \rightarrow U\ B$
6. $W \rightarrow B\ T$
7. $V \rightarrow G\ C\ T$
8. $U \rightarrow E\ G$
9. $T \rightarrow \left\{\begin{matrix} S \\ Z \end{matrix}\right\}\ F$

SPG3 is structure sensitive, since two categories, S and X, are rewritten the same way. This results from the fact that, in DG3, A has two sets of dependents and one set is included in the other. The structure-sensitive rules for C in DG3 produced no additional structure-sensitive rules in SPG3, however, since one of them, C (*), was not processed in step 3 and did not form part of the IRL.

SPG3 may be rewritten as a structure-free grammar in a purely ad hoc way by eliminating the rewriting rule for X and substituting S for the occurrence of X on the right. More generally, it is reasonable to require of the original DG that its rules be designed so that it is possible to write a single rule (schema) assigning dependents to any given category. This requirement is reasonable if the DG is a base component of a transformational grammar whose transformations take care of the eventual order of elements in a sentence. The primary function of the DG's dependency rules is, in this case, only that of listing co-occurring categories in the dependency relations in some canonical order. In DG3, A always occurs with B as dependent. Whenever E is a dependent of A, then either a second B or a C are also dependents, and if a second B is a dependent of A, D is also. This set of conditions is summed up by

$$A\left(B\ *\ \left[\left\{\begin{matrix} B\ D \\ C \end{matrix}\right\}\ E\right]\right).$$

Similarly, B always occurs with E or F as dependents, i.e.,

$$B\left(* \left\{\begin{matrix} E \\ F \end{matrix}\right\}\right);$$

and C occurs with no dependents or else with both G and F, i.e.,

$$C\ (* \ [G \ \ F]).$$

These rules express co-occurrence relationships more directly than the original rules do. Let us assume this constraint and redesign DG3 as DG4. Let the augmented DG4 be:

Axiom: * (A)

Dependency Rules:

1. $A\left(2B \ * \ 1\left[\left\{\begin{matrix} B \quad D \\ C \end{matrix}\right\} \ E\right]\right)$
2. $B\left(* \ 1\left\{\begin{matrix} E \\ F \end{matrix}\right\}\right)$
3. C (* 1[G F])
4. D (*)
5. E (* 1G)
6. F (1A *)
7. G (*)

The IRL is:

$$A^2 \rightarrow B^1 \ A^1_*$$

$$A^1 \rightarrow A^0_* \left[\left\{\begin{matrix} B^1 \quad D^0 \\ C^1 \end{matrix}\right\} \ E^1\right]$$

$$B^1 \rightarrow B^0_* \left\{\begin{matrix} E^1 \\ F^1 \end{matrix}\right\}$$

$$C^1 \rightarrow C^0_* \ [G^0 \ \ F^1]$$

$$E^1 \rightarrow E^0_* \ G^0$$

$$F^1 \rightarrow A^2 \ F^0_*$$

With appropriate substitutions, this becomes SPG4:

Axiom: Z

Rewriting Rules:

1. Z → X Y

2. $Y \rightarrow A \left[\left\{ \begin{matrix} X\ D \\ W \end{matrix} \right\} \ V \right]$

3. $X \rightarrow B \left\{ \begin{matrix} V \\ U \end{matrix} \right\}$

4. $W \rightarrow C\,[G\ \ U]$

5. $V \rightarrow E\ \ G$

6. $U \rightarrow Z\ \ F$

SPG4 is structure free. Furthermore, it has only one axiom, whereas SPG3 had two even though derived from a DG which had only one.

The additional restrictions proposed for DG and SPG in discussions of the results of conversion by these methods are not crucial as far as obtaining systematically corresponding grammars is concerned. Without them, every complete subtree of a D-tree will correspond to a complete subtree of a P-tree over the same string, and every complete subtree of the P-tree will correspond to a connected subtree of a D-tree over the same string. (No formal proof of this was given, but the methods of constructing an IRL make it moderately apparent.) Hays [5] suggests the term *relational correspondence* for this state of affairs. Also there is a systematic relationship between the categories of the two grammars, which the TS makes explicit.

Sometimes that relationship is simple, as in the case of DG4 and SPG4. Given any category of DG4, there is exactly one category of SPG4 from which the same set of strings is derivable. The relationship also holds between the categories of DG1 and SPG1, and between those of DG2 and SPG2. Under these conditions, Hays [5] calls the categories "substantively equivalent". The relationship between the categories of DG3 and SPG3 is less simple. There the set of strings derivable from A of the DG is the union of the set of strings derivable from S and Z of the SPG, and the set derivable from B is the union of the set derivable from W and V.

Hays says that a D-tree and a P-tree *correspond* if they correspond relationally and if the category at the origin of every complete subtree of the D-tree is substantively equivalent to the category labeling the complete subtree of the P-tree related to it. If a DG and an SPG "have the same terminal alphabet, and for every string over that alphabet, every structure attributed by either corresponds to a structure attributed by the other", he calls the two grammars "strongly equivalent". [5, p. 521] We prefer to say that they *correspond substantively*, since relational correspondence is asymmetric and there are always "left-over" SPG categories to which no DG category is substantively equivalent. The weaker relationship exhibited by SPG3 and DG3, where some DG categories are not substantively equivalent to any single SPG category, we have been calling *systematic correspondence*.

Conversion by the methods given here results in systematic correspondence. If the suggested constraints, which appear to be linguistically well-motivated for base components of transformational grammars, are imposed on the form of the source grammar, the target grammar corresponds substantively as well as systematically to the source grammar and both are structure free.

BIBLIOGRAPHY

1. Bar-Hillel, Y., Perles, M., and Shamir, E. "On formal properties of phrase structure grammars," in R. D. Luce, R. Bush, and E. Galanter (eds.), *Readings in Mathematical Psychology*, Vol. II, pp. 75-104. New York, Wiley, 1965.

2. Chomsky, Noam. "On certain formal properties of grammars," in R. D. Luce, R. Bush, and E. Galanter (eds.), *Handbook of Mathematical Psychology*, Vol. II, pp. 323-418. New York, Wiley, 1963.

3. Chomsky, Noam. *Aspects of the theory of syntax*. The M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Mass., 1965.

4. Gaifman, Haim. *Dependency systems and phrase structure systems*. P-2315, The RAND Corporation, Santa Monica, California, May 1961.

5. Hays, David G. "Dependency theory: a formalism and some observations," *Language*, Vol. 40, (Oct.-Dec. 1964), pp. 511-525.

6. Hays, David G. *An annotated bibliography of publications on dependency theory*. RM-4479-PR, The RAND Corporation, Santa Monica, California, March 1965.

7. Kay, Martin. *The tabular parser: A parsing program for phrase structure and dependency*. RM-4933-PR. The RAND Corporation, Santa Monica, California, July 1966.

8. Lees, R. B. *The grammar of English nominalizations*, Supplement to *International Journal of American Linguistics*, 26, No. 3, Part II, 1960.

9. Robinson, Jane. "A dependency-based transformational grammar." IBM Corporation, Yorktown Heights, New York. (Forthcoming, 1967)

10. Rosenbaum, Peter S. "English Grammar II." IBM Corporation, Yorktown Heights, New York. (Forthcoming, 1967)

11. Zwicky, A. M., Hall, B. C., Fraser, J. B., Geis, M. L., Mintz, J. W., Isard, S., and Peters, P. S. "English preprocessor manual." Informations System Language Studies Number Seven, SR-132, The MITRE Corporation, December 1964.

17

The Elimination of Grammatical Restrictions in a String Grammar of English

M. Salkoff and N. Sager

Institute for Computer Research in the Humanities
New York University, New York

1. Summary of String Theory

In writing a grammar of a natural language, one is faced with the problem of expressing grammatical dependencies. For example, in the sentence form $\underline{N\ V\ N}$ (N, noun: V, verb), the subject $\underline{N}$ and the verb $\underline{V}$ must agree in number: The boy eats the meat; $\not\exists$ The boys eats the meat. Or, in the sequence $\underline{Q\ N_1\ P\ N_2}$ (Q a number; P, preposition), e.g., five feet in length, N_1 and N_2 are of particular subclasses: $\not\exists$ five feet in beauty. One of the theories of linguistic structure which is particularly relevant to this problem is linguistic string analysis [1]. In this theory, the major syntactic structures of English are stated as a set of elementary strings (a string is a sequence of word categories, e.g., $\underline{N\ V\ N}$, $\underline{N\ V\ P\ N}$, etc.). Each sentence of the language consists of one elementary sentence (its center string) plus zero or more elementary adjunct strings which are adjoined either to the right or left or in place of particular elements of other elementary strings in the sentence.

The elementary strings can be grouped into classes according to how and where they can be inserted into other strings. If $Y = X_1 X_2 \ldots X_n$ is an elementary string, X ranging over the category symbols, the following classes of strings are defined:

ℓ_X	left adjuncts of X: adjoined to a string Y to the left of X in Y, or to the left of an ℓ_X adjoined to Y in this manner.
r_X	right adjuncts of X: adjoined to a string Y to the right of X in Y, or to the right of an r_X adjoined to Y in this manner.
$n_X^{\cdot}$	replacement strings of X: adjoined to a string Y, replacing X in Y.
s_Y	sentences adjuncts of the string Y, adjoined to the left of X_1 or after X_i in Y ($1 \leq i \leq n$), or to the right of an s_Y adjoined to Y in this manner.
$c_{Y,i}$	conjunctional strings of Y, conjoined after X_i in Y ($1 \leq i \leq n$), or to the right of a $c_{Y,i}$ adjoined to Y in this manner.
z	center strings, not adjoined to any string.

These string-class definitions, with various restrictions on the repetition and order of members of the classes, constitute rules of combination on the elementary strings to form sentences.

Roughly speaking, a center string is the skeleton of a sentence and the adjuncts are modifiers. An example of a left adjunct of N is the adjective green in the green blackboard. A right adjunct of N is the clause whom we met in the man whom we met. A replacement formula of N is, for example, what he said in the sentence What he said was interesting. The same sentence with a noun instead of a noun replacement string might be The lecture was interesting. Examples of sentence adjuncts are in general, at this time, since he left. The c strings have coordinating conjunctions at their head. An example is but left in He was here but left. Examples of center strings are He understood and also We wondered whether he understood.

The grammatical dependencies are expressed by restrictions on the strings as to the word subcategories which can occur together in a string or in strings related by the rules of combination. Thus, in the center string N_1 V N_2, the grammatical dependency mentioned above is formulated by the restriction: if N_1 is plural, then V does not carry the singular morpheme -s. The string grammar with restrictions gives a compact representation of the linguistic data of a language, and provides a framework within which it is relatively simple to incorporate more linguistic refinement, i.e., more detailed restrictions.

One may ask whether it is possible to write such a string grammar without any restrictions at all, i.e., to express the grammatical dependencies (restrictions) in the syntactic structures themselves. In the resulting restrictionless grammar, any elements which are related by a grammatical dependency will be elements of the same elementary string. No grammatical relations, other than those given by the simple rule of string combination, obtain between two strings of a sentence. The result of this paper is to demonstrate that such a restrictionless grammar can be written [4].

In order to obtain a restrictionless form of a string grammar of English, we take as a point of departure the grammar used by the computer program for string decomposition of sentences, developed at the University of Pennsylvania [2,3]. This grammar is somewhat more detailed than the sketch of an English string grammar in [1]. A summary of the form of the computer grammar is presented below in section 2. In section 3 we show how the restrictions can be eliminated from the grammar.

An example of a typical output obtained for a short sentence from a text of a medical abstract is shown in Figs. 1 and 2. The decomposition of the sentence into a sequence of nested strings is indicated in the output by the numbering of the strings. As indicated in line 1., the sentence consists of the two assertion centers in lines 2. and 4., conjoined by and. The line 3.

contains a sentence adjunct (thus) on the assertion center as a whole. The assertion center 2. is of the form N V A : Spikes would be effective. The noun spikes has a left adjunct (such enhanced) in line 5., as indicated by the appearance of 5. to the left of spikes. The object effective has a left adjunct (more) in line 6. and a right adjunct in line 7. In the same way, each of the elements of the adjunct strings may have its own left and right adjuncts. Line 10. contains an assertion center in which the subject and the modal verb (would) have been zeroed. This zeroing is indicated in the output by printing the zeroed element in parentheses.

The difference between the two analyses in Figs. 1 and 2 lies in the decomposition of the sequence in initiating synaptic action. In the first analysis (Fig. 1), this sequence is taken as a P N right adjunct on effective, where initiating synaptic is a left adjunct (on action) of the form of a repeated adjective (parallel to escaping toxic in the sequence in escaping toxic gases). In the second analysis (Fig. 2), this same sequence is taken as a P Ving right adjunct of effective, where initiating is the Ving, and synaptic action is the object of initiating.

2. The Computer String Grammar.

In representing the string grammar in the computer, a generalized grammar string is used[5] which is defined as

(1) $Y = Y_1 / Y_2 / \ldots / Y_n$

where

(2) $Y_i = Y_{i1} Y_{i2} \ldots Y_{im}$

and

(3) $Y_{ij} = Y'$ where Y' is a grammar string like Y.

This system of nested grammar strings terminates when one of the grammar strings is equal to an atomic string (one of the word-category symbols). The Y_i are called the options of Y, and each option Y_i consists of the elements Y_{ij}.

Not every option of a grammar string Y will be well-formed each time the sentence analysis program finds an instance of Y in the sentence being analyzed. Associated with each option Y_i is a series of zero or more tests, called restrictions. If R_i, is the set of tests associated with Y_i then the grammar string Y can be written:

(4) $Y = R_1Y_1 / R_2Y_2 / \ldots / R_nY_n$

A restriction is a test (which will be described below) so written that if it does not give a positive result its attached option may not be chosen.

All of the restrictions in the grammar fall into two types:

Type A: The restrictions of type A enable one to avoid defining many similar related sets of grammar strings. The options of the grammar string Y have been chosen so that Y represents a group of strings which have related linguistic properties. This allows the grammar to be written very compactly, and each grammar string can be formulated as best suits the linguistic data. However, when a grammar string Y appears as a Y'_{ij} of some other string Y', some of the options of Y may lead to non-wellformed sequences. In order to retain the group of options of Y and yet not allow non-wellformed sequences wherever options of Y which would have that effect are used, we attach a restriction of type A to those options of Y.

For example, let Y be

(5) $Y = Y_1 \ / \ R^a \ Y_2 \ / \ . \ . \ .$

where

(6) Y_1 = which Σ V (e.g., which he chose)

and

Y_2 = what Σ V (e.g., what he chose)

Then Y can appear in the subject Σ of the linguistic center string C1:

(7) C1 = Σ V Ω

This yields Which he chose was important; What he chose was important.

As it is defined here, Y can also be used to represent the wh-clauses in the right adjuncts of the noun:

(8) $Y' = r_N = . \ . \ . \ / \ Y \ / \ . \ .$

but in r_N only the which option of Y gives wellformed sequences:

∃ the book which he chose

∄ the book what he chose

Hence a restriction R^a is attached to the what option of Y (eq. 5) whose effect is to prevent that option from being used in r_N.

Type B: With some given set of rather broadly defined major categories (noun, verb, adjective, etc.) it is always possible to express more detailed linguistic relations by defining sub-categories of the major categories. These relations then appear as constraints on how the sub-categories may appear together in the grammar strings Y.

If some element Y_{ij} of Y_i is an atomic string (hence a word-category symbol) representing some major category, say C, then R^b may exclude the sub-category C_j as value of Y_{ij} if some other element Y_{ik} of Y_i has the value C_k. Y_{ik} may also be a grammar string, in which case R^b may exclude a particular option of Y_{ik} when Y_{ij} has value C_j.

The restrictions R^b may be classified into three kinds:

(a) Between elements of some string Y_i where the Y_{ij} correspond to elements of a linguistic string.

For example,

> A noun in the sub-category singular cannot appear with a verb in the sub-category plural. $\not\exists$ The man agree.
>
> Only a certain sub-category of adjective can appear in the sentence adjunct P A : in general, in particular, $\not\exists$ in happy.

(b) Between a Y_{ij} and a Y_{ik} where Y_{ij} corresponds to an element of a linguistic string and Y_{ik} corresponds to a set of adjuncts of that element. For example,

> In r_N, the string to V Ω cannot adjoin a noun of sub-category N_2 (proper names): the man to do the job $\not\exists$ John to do the job.

Only a certain adjective sub-category (e.g., *present*, *available*) can appear in r_N without any left or right adjunct of its own: *the people present* ; $\nexists$ *the people happy*.

(c) Between Y_{ij} and Y_{ik} , where one corresponds to an element of a linguistic string and the other corresponds to an adjunct set which can repeat itself, i.e., which allows 2 or more adjuncts on the same linguistic element. These restrictions enable one to express the ordering among adjuncts in some adjunct sets. For example,

> *Q* (quantifier) and *A* (adjective) are both in the set ℓ_N , the left adjuncts of the noun. However, *Q* can precede *A* but *A* cannot precede *Q* when both are adjuncts of the same *N* in a sentence: $\exists$ *Q A N* e.g., *five green books* , but $\nexists$ *A Q N* e.g., *green five books*.

The string grammar defined by eqs. 1-3, together with the atomic strings (word-category symbols) have the form of a BNF definition. The system with eq. 4, however, departs from a BNF definition in two important respects:

(a) it contains restrictions (tests) on the options of a definition;

(b) the atomic strings (word-categories) of the grammar have sub-classifications.

With the elimination of the restrictions, the computer grammar will again have the form of a BNF definition.

3. Elimination of the Restrictions

The restrictionless string grammar is obtained from the grammar described above by the methods of (A) and (B) below. Initially (in this paper), conjunctional strings have not been included in the restrictionless grammar. We estimate that the addition of conjunctional strings will increase the size of the restrictionless grammar by a factor of about 5.

(A) The linguistic strings represented in the computer grammar are reformulated in accordance with the following requirement. Given any utterance of a language containing A . . . B . . . , where a grammatical dependency obtains between A and B , the elementary strings of a restrictionless string grammar are defined so that A and B appear together in the same linguistic string, and any iterable sequence between A and B is an adjunct of that string. Iterable sequences of the type seemed to begin to in It seemed to begin to surprise him that we worked seriously , or is said to be known to in It is said to be known to surprise him that we worked seriously are analyzed as adjuncts. If we place such sequences among the left adjuncts of the verb, ℓ_v , then the sentences above can be put in the form

(9) It ℓ_v surprise him that we worked seriously

ℓ_v = *seemed to begin to* ; *is said to be known to* ; etc.

However, when the adjunct ℓ_v takes on the value zero (as can all adjuncts, by definition), then (9) above becomes the non-grammatical sequence *It surprise him that we worked seriously*. This happens because the first verb of ℓ_v (*seemed* or *is*) carries the tense morpheme, and the latter disappears when $\ell_v = 0$. We separate the tense morpheme from the verb, and place it in the center string as one of the required elements.

(10) $Cl = \Sigma\ t\ \ell_v\ V\ \Omega$; $t = 0$ / -*s* / -*ed* / *will*, *can*, ...

This formulation of the assertion center string Cl (10), in which the tense morpheme is an independent element and iterable sequences are taken as adjuncts, is necessary in order to preserve, for example, the dependence between the particle *it* and the succeeding sequence *surprises him that we worked seriously*: $\nexists$ *The book surprises him that we worked seriously*. In the grammar which includes restrictions, this formulation is not necessary because this dependence can be checked by a restriction.

(B) Turning to the computer form of the grammar, all the restrictions of the grammar are eliminated either by defining new grammar strings (for the elimination of the restrictions R^a) or by replacing the general word-categories by the particular subclasses of those categories which are required by the restriction (to eliminate R^b). The application of this procedure increases the number of strings in the grammar, of course.

The restrictions R^a can be eliminated in the following manner. Suppose the option Y_i of Y has a restriction R^a on it which prevents it from being chosen in Y' (Y is a Y'_{ij} of Y'). Then define a new grammar string Y* which contains all the options of Y but Y_i :

(15) $Y^* = Y_1 / Y_2 / \ldots / Y_{i-1} / Y_{i+1} / \ldots / Y_n$

Then the new grammar string Y^* replaces Y in Y'. Thus, in the example of R^a on p. 5, the string Y^* = which Σ t ℓ_V V / .... (in the modified treatment of tense and iterable sequences) would replace Y in r_N.

The restrictions R^b are eliminated in a different way, according to the types described on p. 6.

(a) New strings must be written in which only the wellformed sequences of subcategories appear. In the example of subject-verb agreement, the original Y_i (Y_i = Cl) must be replaced by two options:

$$Cl = N\ t\ V\ \Omega \rightarrow N_s\ t\ V_s\ \Omega / N_p\ t\ V_p\ \Omega$$

where N_s and N_p are singular and plural nouns, V_s and V_p singular and plural verbs.

(b) If an element of a particular subcategory, say A_i, can take only a subset of the adjuncts r_A, then a new adjunct string r_{Ai} is defined. It contains those options of r_A which can appear only with A_i plus all the options of r_A which are common to all the sub-categories of A. When this has been done for all A_i having some particular behavior with respect to r_A, all the remaining sub-categories of A will have a common adjunct string r_a :

$$A\ r_A \rightarrow A_1 r_{A1} / A_3 r_{A3} / \ldots / A_2 r_a / A_4 r_a / \ldots$$

As many new sets r_{Ai} must be defined as there were special sub-categories of A. A similar argument holds for ℓ_A and other adjunct sets which depend on A.

(c) A new element corresponding to the adjunct set must be defined in which the adjuncts appear correctly ordered with respect to each other, and each one must be able to take on the value zero.

This procedure for eliminating restrictions is also the algorithm for introducing further grammatical refinements into the restrictionless grammar. Such a general procedure can be formulated because of an essential property of a string grammar: In terms of linguistic (elementary) strings, all restrictions are either a) between elements of a string, or b) between an element of the string and its adjunct, or c) between related adjuncts of the same string. Further, there is no problem with discontinuous elements in a string grammar: all elements which depend in some way on each other grammatically appear in the same string or in strings which are contiguous by adjunction.

The cost of the elimination of all restrictions in this way is about an order of magnitude increase in the number of strings of the grammar. Instead of about 200 strings of the computer grammar, the grammar presented here has about 2000 strings. It is interesting that the increase in the size of the grammar is not greater than roughly one order of magnitude. This suggests that there may be practical applications for such a grammar, e.g. in a program designed to carry out all analyses of a sentence in real time. Also, since the restrictionless grammar is equivalent to a B.N.F. grammar of English, it may prove useful in adding English-language features to programming languages which are written in B.N.F.

Figure 1

```
SENTENCE NEUR-18 . SUCH ENHANCED SPIKES WOULD BE MORE EFFECTIVE
           IN INITIATING SYNAPTIC ACTION AND THUS BE RESPONSIBLE
           FOR THE OBSERVED POST-TETANIC POTENTIATION .

    PARSE     01
 1. SENTENCE      = INTRODUCER CENTER AND        END MARK
                              2.    AND  3.  4. .

 2. C1 ASSERTION  = * SUBJECT    * VERB     * OBJECT             RV *
                     5. SPIKES   WOULD BE    6. EFFECTIVE  7.

 3. ACVERB        = ADVERB
                    THUS

 4. CONJUNCTION   = CENTER
                    10.

 5. LN            = ARTICLE QUANTIFIER ADJECTIVE TYPE-NS' NOUN
                            SUCH       ENHANCED

 6. ACVERB        = ACVERB
                    MORE

 7. P N           = LP PREPOSITION N
                       IN         11. ACTION

10. C1 ASSERTION  = * SUBJECT       * VERB       * OBJECT          RV *
                     ( 5. SPIKES )   (WOULD ) BE   RESPONSIBLE 12.

11. LN            = ARTICLE QUANTIFIER ADJECTIVE   '     TYPE-NS NOUN
                                       INITIATING SYNAPTIC

12. P N           = LP PREPOSITICN N
                       FOR        13. POTENTIATION

13. LN            = ARTICLE QUANTIFIER ADJECTIVE             TYPE-NS NOUN
                    THE        '       OBSERVED POST-TETANIC
```

Figure 2

```
SENTENCE NEUR-18 . SUCH ENHANCED SPIKES WOULD BE MORE EFFECTIVE
           IN INITIATING SYNAPTIC ACTION AND THUS BE RESPONSIBLE
           FOR THE OBSERVED POST-TETANIC POTENTIATION .

    PARSE      02
 1. SENTENCE          = INTRODUCER CENTER AND         END MARK
                                      2.    AND  3.  4.  .

 2. C1 ASSERTION      = * SUBJECT    * VERB      * OBJECT            RV *
                          5. SPIKES   WOULD BE    6. EFFECTIVE  7.

 3. ADVERB            = ADVERB
                        THUS

 4. CONJUNCTION       = CENTER
                        10.

 5. LN                = ARTICLE QUANTIFIER ADJECTIVE TYPE-NS NOUN
                                SUCH       ENHANCED

 6. ADVERB            = ADVERB
                        MORE

 7. P NS VING(OF) O = PREPOSITION SN
                        IN           INITIATING 11. ACTION

10. C1 ASSERTION      = * SUBJECT         * VERB          * OBJECT           RV *
                        ( 5. SPIKES )    (WOULD ) BE    RESPONSIBLE 12.

11. LN                = ARTICLE QUANTIFIER ADJECTIVE TYPE-NS NOUN
                                           SYNAPTIC

12. P N               = LP PREPOSITION N
                           FOR          13. POTENTIATION

13. LN                = ARTICLE QUANTIFIER ADJECTIVE              TYPE-NS NOUN
                        THE                OBSERVED POST-TETANIC

NO MORE PARSES
```

REFERENCES

1. Harris, Z. S., String Analysis of Sentence Structure, Papers on Formal Linguistics, No. 1, Mouton and Co., The Hague, 1962.

2. Sager, N., Salkoff, M., Morris, J., and Raze, C., Report on the String Analysis Programs, Department of Linguistics, University of Pennsylvania, March 1966.

3. Sager, N., "Syntactic Analysis of Natural Language", Advances in Computers (Alt, F. and Rubinoff, M., eds.), vol. 8, pp. 153-188. Academic Press, New York, 1967.

4. This problem was suggested by Professor J. Schwartz of the Courant Institute of Mathematical Sciences, New York University.

5. The option Y_i here corresponds to the linguistic string Y of the previous section. The symbol / separates the options of a string definition.

The Application of FORTRAN to Automatic Translation

Paul O. Samuelsdorff

1. Introduction

A multitude of problem-oriented programming languages are being created in order to facilitate programming in various fields. These languages have two advantages for non-programmers who need a computer for solving their problems: 1. they are not forced to spend much time on learning a complicated machine language; 2. they do not necessarily have to alter their programs when they are obliged to use a different machine. These advantages seem to justify the labour of writing a multitude of compilers that translate each problem-oriented language into the machine language of each machine.

The existence of various problem-oriented languages, however, makes it difficult for the users of computers in various fields to exchange their experience. In addition there is the danger that a certain machine possesses only a limited number of compilers, and that therefore a program may have to be rewritten when for some reason or other a different machine has to be used. The question therefore arises whether it is not possible to have the advantages of problem-oriented languages without multiplying their number.

This paper wants to show that one of the most widespread problem-oriented languages, originally created for mathematicians, may efficiently be used for non-numerical data-processing, if the necessary subroutines are available.

Another object of this paper is to give a concrete picture of how linguistic problems may be handled by the computer to those readers who have only a theoretical knowledge of data-processing. For the benefit of these readers a few definitions will be given, so that the examples will be understood by everybody.

bit = the smalles unit of information that may be stored in the computer, having one of two values symbolized by "1" or "0".

character = a letter of the alphabet, a decimal digit, or one of 11 special signs including the blank. A character is represented in the computer used for our examples (IBM 7094) by a combination of 6 bits.

machine-word = a string of six characters (represented in the computer by a combination of 36 bits).

location = the place in the computer where a machine-word is stored.

array = a combination of locations treated as a unit.

subroutine = a program that may be called up by another program for performing a specific job.

function = a subroutine that in addition to handling a certain job leaves a definite result in a certain location.

IF-statement = an instruction of the program that enables a branching to one of three different statements in the program, depending on whether the value in the parentheses of this statement is negative, zero or positive. Its format is "IF (x) l,m,n", where l,m and n are the numbers of the statements to which the program may branch.

The application of a mathematically oriented programming language to linguistic problems will be exemplified by a Hebrew-English translation project written in FORTRAN with the aid of subroutines programmed at the Deutsches Rechenzentrum in Darmstadt. These subroutines handle data smaller than the machine-word (single bits, single characters and their concatenation), the non-numerical machine-word and data larger than the machine-word.

We shall now explain the subroutines used in our project and give examples of their application. In order to avoid a repeated explanation of the variables used in the formats of the subroutines, we introduce the following conventions.

M = a positive integer not greater than 36, referring to one of the 36 bits of a machine-word.

WORD = the location of a machine-word to be handled by the subroutine. If two machine-words are to be handled, we distinguish between WORD1 and WORD2.

BLOCK = the location of the first machine-word of an array containing the characters or the string to be handled by the subroutine. If two arrays are involved, we distinguish between BLOCK1 and BLOCK2.

NA = a positive integer indicating the position of the first character of a string in the array BLOCK to be handled by the subroutine. If two strings are to be handled, we distinguish between NA1 and NA2.

N = an integer, usually positive, indicating the number of characters to be treated by the subroutine. N will be negative if we count the characters from NA to the left.

TABLE = an array containing characters to be used by a subroutine. This table is prepared before the subroutine is called.

The subroutines will be described in the order of the units they handle, starting from the smallest units.

2. Bits

2.1. PAKR

This routine facilitates the coding of grammatical information. Each bit of a 36-bit machine-word represents a certain kind of grammatical information. Its value is "1" when this information applies to the Hebrew word it is coordinated to and "0" when this information does not apply. The 36 bits of one machine-word, stored in the array CODE, represent the following information.

1 = definite noun
2 = indefinite noun
3 = construct state
4 = pronoun
5 = prepositional phrase
6 = preposition
7 = adverb
8 = numeral
9 = indefinite adjective
10 = definite adjective
11 = indefinite participle
12 = definite participle
13 = ZH
14 = AT
15 = AUTU, AUTH, AUTM, AUTN
16 = KI, AM, KAWR
17 = GM, RQ
18 = LA
19 = IW
20 = verb
21 = infinitive
22 = comma
23 = KL
24 = present
25 = past
26 = future
27 = copula
28 = WL
29 = W
30 = U
31 = masculine singular
32 = feminine singular
33 = masculine plural
34 = feminine plural
35 = no indefinite article
36 = no definite article

(The words in capital letters are Hebrew words that need special treatment.)

It would be very complicated to work out the **figure** to be stored in a certain location for each possible combination of grammatical information. Let us take as an example a masculine singular verb in the past tense. The machine-word coordinated to this verb will have a "1" in bit-positions 20, 25, and 31, and a "0" in all other bit-positions: 000000000000000000010000100000100000. This is the binary equivalent of the decimal number 69 664. Instead of working out this figure, we code a "1" into the 20th, 25th and 31st column of a punched card and leave the other columns blank. The first 36 columns of the card are then read into an array of 36 locations. The 36 machine-words of this array are subsequently packed with the aid of the subroutine PAKR into one location (containing 36 bits), the ones and blanks of the 36 columns on the card resulting in ones and zeros in the respective bits of one machine-word.

2.2. KBIT

This subroutine is a function and identifies single bits. If we want to find out whether a word possesses a certain grammatical feature, we have to ask whether the value of the respective bit of the machine-word that contains the desired grammatical information is "1" or "0".

Format: J = KBIT(WORD,M)

The result stored in the location J will be "1" or "0", depending on the value of the respective bit.

The subroutine KBIT enables the control to cause a branching to the specific part of the program needed for the treatment of a member of a certain word-class. A computed GOTO after the dictionary lookup handles this process very efficiently, as seen in the following example.

```
79   DO     90    L = 1,23
     IF (KBIT(CODE(N),L))  999,90,80
80   GO TO (100,200,300,400,500,600,1000,75,920,910,1200,2000,
    13000,4000,5000,1000,75,1000,1000,2200,1000,1000,700),L
90   CONTINUE
```

The value of the variable L must be computed before the GOTO statement is reached. It must be a positive integer not greater than the number of statement-numbers in the parentheses of the GOTO statement, since it determines which statement of the program is to be executed next. L is computed in statement 79. It assumes the values from 1 to 23 as long as the program reaches statement 90. The IF-statement between statement 79 and statement 80 causes the program to branch to statement 90 as long as the Lth bit of the machine-word containing the grammatical information of the Nth word of the sentence equals zero. As soon as the Lth bit equals one, the program will branch to statement 80. (999 is a dummy statement number, since the result of KBIT will never be negative.) Since we are examining the machine-word found in the dictionary together with the Hebrew word we were looking up, statement 80 will cause a branching to that part of the program that handles members of the word-class this Hebrew word belongs to.

An IF-statement using the subroutine KBIT may also enable the program to branch conditionally during the syntactic analysis or the translation process and thus make the solution of ambiguities possible. Specific parts of the program for the solution of ambiguities are reached by statements examining the bits of machine-words located in the array SCODE. For instance for ambiguous verbs the statement will be:"2200 IF (KBIT(SCODE(N),1)) 999,2210,2500". The part of the program solving the ambiguity of the verb starts at statement 2500. The other "1"-bits of SCODE, coded with a verb or a noun, indicate how a preposition is to be translated.

2.3. LBIT

This subroutine permits the conversion of "0"-bits into "1"-bits and vice versa. Its format is similar to that of KBIT, namely

J = LBIT(WORD,M)

The result stored at J will be "1" when "0" has been replaced by "1" and "0" when "1" has been replaced by "0".

LBIT is used for storing or deleting grammatical information during the analysis process.

An example is the indication of definite nouns or prepositional phrases during the dictionary lookup. Since in Hebrew the definite article and some of the prepositions consist of one letter that is attached to the following noun, this letter has to be stripped before the word can be found in the dictionary. We use the subroutine LBIT during the dictionary lookup for storing "1" in the first bit-position of CODE(N)and deleting the "1" in the second bit-position if the Nth word of the sentence is a noun preceded by the definite article, or "1" is stored in the fifth bit-position, if the Nth word of the sentence is a noun preceded by a preposition.

3. Single characters

3.1. IFFP (find first position)

This subroutine makes the identification of single characters possible.

Format: J = IFFP(BLOCK,NA,N,TABLE,IND)

This function finds the first position of a character in the array BLOCK which is equal to a character of TABLE if IND = 1 or unequal to any of the characters of TABLE if IND = 0. The search starts from the NAth character of the array. It goes to the right if N is positive and to the left if N is negative.

One of the characters we frequently want to identify is the blank, in order to find word boundaries. This subroutine is used, for instance, when we want to insert an adjective between a preposition and the following noun. We have to look for the blank after the preposition in order to insert the adjective in the right place. An example for a negative N will be given in section 6.3.

3.2. SEPO (seek positions)

This subroutine traces the positions of occurrence of a character (or a member of a set of characters) stored in a table.

Format: CALL SEPO (BLOCK,NA,N,TABLE,NSP,NFIND)

The subroutine SEPO examines N characters in the array BLOCK, starting from the NAth character. The number of characters found to be equal to the characters stored in TABLE is stored in the location NFIND. The actual positions of these NFIND characters are stored in the array NSP.

If the only character stored in the table is the blank, dictionary lookup will be very convenient, since the occurrence of word boundaries may be stored for a whole text. NSP(I) will contain the position of the blank after the Ith word of the text. The next word to be processed will therefore easily be found. Its first character will be in the first position after the one indicated by the contents of NSP(I) and its length will be NSP(I+1) - NSP(I) - 1.

3.3. ICPC (compare character)

This subroutine compares a character with a set of characters stored in a table.

Format: J = ICPC(BLOCK,NA,TABLE,NTAB)

NTAB is a positive integer stating with how many characters of TABLE we want to compare the NAth character of the array.

The result of this function, stored in J, is zero if the NAth character is unequal to any of the characters of the table, otherwise it is a positive integer not greater than NTAB, indicating that the NAth character of the array BLOCK is equal to the Jth character of TABLE.

In the following example we see how this subroutine is used for choosing between the English indefinite articles "a" (WORD(1)) and "an" (WORD(5)).

```
1476 K = ICPC (EWD(1,N),1,VOC,5)
1477 IF (K-1)   1905,1910,1915
1905 ENG(1,I) = WORD(1)
     I = I+1
     GO TO 1480
1910 K = ICPC (EWD(1,N),3,VOC,5)
     IF (K)   1905,1915,1905
     ENG(1,I) = WORD(5)
     I = I+1
     GO TO 1480
```

EWD is the array where the English words were brought after the dictionary lookup and ENG where they are brought during the analysis. The first five locations of the table VOC contain the vowels U, O, I, E, A. If the first letter of the noun in question, stored in EWD(1,N), starts with a consonant, K in statement 1476 will be zero, and K-1 in statement 1477 will be negative. If the first letter is U, K-1 =0, and if the first letter is any other vowel, K-1 will be positive. That means, that for any first letter of the noun, except U, statement 1477 decides whether "a" or "an" will be chosen. In the case of U the choice depends on the third letter of the noun, found in statement 1910. "A" will be chosen if the third letter is a vowel, and "an" will be chosen if it is a consonant. This routine will, of course, still have to be refined for the case when the prefix "un" is followed by a vowel.

4. Machine-word parts

4.1. CSWO (compose word)

This subroutine, composing a machine-word from two parts of two other machine-words, facilitates the insertion of short words like prepositions or the definite article.

Format: S = CSWO (WORD1,WORD2,K)

As a result of this function the first K characters of the machine-word in location WORD1 is composed with the last 6 - K characters of the machine-word in location WORD2. The resulting machine-word is stored in location S. S may be identical with WORD1 or WORD2. We see an example of the application of this subroutine in the part of the program that translates a date into English.

```
375 EWD(1,IL) = CSWO (WORD(30),EWD(1,IL),2)
    K = IL-1
    CALL SHA2(EWD(1,K),1,32,4,1)
    EWD(1,K) = CSWO (WORD(40),EWD(1,K),3)
    CALL SHA2 (EWD(1,K),1,33,3,1)
    GO TO 330
```

In statement 375 the preposition "of" (WORD(30)), chosen in this case as the translation of the Hebrew prepositional prefix B, is composed with the first part of the name of the month which follows it. The statements following statement 375 translate the Hebrew B before the day of the month by "on the" (WORD(32) and WORD(40)). The shifting subroutine SHA2 will be explained in section 6.2.

5. Machine-words

5.1. CPWO (compare words)

A special subroutine for comparing non-numerical machine-words is necessary for ignoring the sign-bit. When we make a mathematical comparison between two machine-words (subtracting one word from the other) the first bit is interpreted as a sign (0 = + and 1 = -). This will give a wrong result when we compare a word starting with a letter between A and I with a word starting with a letter between J and Z, since the former have "0" in their first bit-position, while the latter have "1".

In non-numerical data-processing we therefore have to regard the machine-word as an entity of six characters, where the first bit of the first character is part of the binary number representing the whole machine-word.

Format: S = CPWO(WORD1,WORD2)

The machine-words at the two locations WORD1 and WORD2 are compared. S = 1 if the contents of WORD1 is greater than the contents of WORD2, S = 0 if the two machine-words are equal, and S = -1 if the contents of WORD1 is smaller than the contents of WORD2.

6. Units larger than the machine-word

6.1. CPST (compare strings)

This routine differs from CPWO in enabling us to define the number of characters (N) that we regard as the entity to be compared.

Format: CPST(BLOCK1,NA1,BLOCK2,NA2,N)

The result is similar to that of the comparison of two machine-words: S = 1 if the binary number representing the string in BLOCK1 (starting from NA1) is greater than the binary number representing the string in BLOCK2 (starting from NA2), S = 0 if the two strings are equal, and S = -1 if the binary number representing the string in BLOCK1 is smaller than the binary number representing the string in BLOCK2.

We shall see an example of this subroutine in the dictionary lookup, section 6.5., where the words from the text will be compared to the words in the dictionary.

6.2. SHA2 (shift in array)

This subroutine enables us to shift strings of variable length.

Format: CALL SHA2 (BLOCK,NA,N,NSHIFT,IND)

NSHIFT is the number of positions we want to shift a string of N characters, starting from the NAth character in the array BLOCK. NSHIFT may be a positive or a negative integer, depending on whether we want to shift to the right or to the left. IND = 1 if we want to fill the vacant positions by blanks and IND = 0 if we want to fill them by zeros. We saw two examples of this subroutine in section 4.1. Since we reserved six machine-words for one English word, N+NSHIFT = 36 (6x6 characters).

6.3. ICSS (compose strings)

With the aid of this subroutine we may bring a string of variable length (N) from one array (BLOCK1) to another (BLOCK2), starting from any position in the two arrays (NA1 and NA2).

Format: J = ICSS(BLOCK1,NA1,N,BLOCK2,NA2)

The result of this function will be J = NA2+N.

This routine, combined with IFFP (section 3.1.), is very useful in bringing the processed English sentence,word for word, into the output area.

```
5060 NA2 = 1
     DO   5075   L = 1,I
     N = IFFP (ENG(1,L),36,-36,TAB,0)+1
     IF (N-1)   5075,5075,5070
5070 NA2 = ICSS (ENG(1,L),1,N,OUT,NA2)
5075 CONTINUE
```

I is the number of arrays, each consisting of six locations, that contain the English words after the analysis process. IFFP has to look for the first non-blank, since the English equivalent of a Hebrew word may be a word group. For the same reason we have to start the search from the 36th character, going left. We add 1 to the result of the function, so that the English word (or word group) is stored with an additional blank and the result (NA2) of the function ICSS may serve as a parameter of the same function in the next round. The IF-statement before 5070 is a safeguard against the English word being blank.

6.4. A subroutine changing the word order was added to the existing subroutines. Since in Hebrew the adjective always follows the noun it qualifies, we have to change the word order in English.

Format: CALL ADT(N)

N is a positive integer indicating that the adjective is the Nth word in the sentence. It changes position (together with its grammar codes) with the (N-1)th word. This subroutine was entirely written in FORTRAN, without using subroutines.

6.5. The following subroutine was written for the dictionary lookup of the translation project. The words in the dictionary are alphabetically ordered, with the exception of the blank, that stands between R and S, because of the value of its binary equivalent. For this reasen, the blank following the word in the text is added before the word is compared with the dictionary entries (HW).

The following parameters are used by the subroutine:

NA = The position of the first letter of the word in the text that is looked up in the dictionary.

NZ = The number of characters of the word.

IL = The ordinal number of the word in the sentence.

M = The number of entries in the dictionary.

The following two constants are set up in the subroutine:

N = the first power of 2 greater than M (in our case 1024)

J = $_2$Log N (in our case 10)

```
      SUBROUTINE LOOKUP (NA,NZ,IL,M)
      DIMENSION HW(4,1024),HC(1024),HSC(1024),EW(6,1024),EC(1024),
     1TEXT(200),CODE(50),SCODE(50),EWD(6,50),ECODE(50),TESTWD(2)
      COMMON  TEXT,HW,CODE,SCODE,EWD,ECODE,HC,HSC,EW,EC,TESTWD
      TESTWD(1) = 6H000000
      J = 10
      N = 1024
      NI = N/2
 535  N = N-NI
 540  J = J-1
      IF (J)   590,544,544
 544  NI = NI/2
 545  IF (CPST (TEXT,NA,HW(1,N),NZ+1))   535,575,555
 555  N = N+NI
 560  IF (M-N)   565,540,540
 565  N = N-NI
      NI = NI/2
      N = N+NI
      J = J-1
      IF (J)   590,570,570
 570  IF (M-N) 565,544,544
 575  CODE(IL) = HC(N)
      SCODE(IL) = HSC(N)
      DO   580   K = 1,6
 580  EWD(K,IL) = EW(K,N)
      ECODE(IL) = EC(N)
      J = LBIT(TESTWD(1),1)
 590  RETURN
      END
```

The lookup is done by the logarithmic method. We jump into the middle of the array reserved for the dictionary and exclude the lower half of the dictionary from further search if the textword is greater than the word of the dictionary that it was compared with, and exclude the upper half if the word was smaller than the word in the dictionary. This process of halving the searched area is repeated till either the word is found or J, which is decreased by one before each search, is equal to zero. That means, we never have to compare more than ten times to find a word in the dictionary or to know that it cannot be found. Statements 560 and 570 prevent us from searching in the area between M and N. When the word is found, the grammatical information from the dictionary and the English translation of the Hebrew word is delivered to the main program and the first bit-position of TESTWD(1) is set "1".

If the word is not found in the dictionary, the program proceeds as follows. If the first letter is one of the possible Hebrew prefixes (found by ICPC), the word is looked up again, with the first letter stripped and NZ diminished by one. The result of the function ICPC is used by a computed GOTO for a branching to that part of the program that will prepare the insertion of the definite article or the solution of the ambiguity of the preposition.

If the first letter is none of the possible prefixes and the word is not found in the dictionary, the program will use the Hebrew original in the translation and print out a message that the word was not found.

7. Conclusion

Apart from the subroutines described above, there are additional subroutines that facilitate information retrieval and other linguistic data-processing. Statistical linguistics may obviously be efficiently treated by a mathematically oriented programming language.

This paper is not supposed to be a plea for FORTRAN, but for a universally applicable problem-oriented language, which will make the progress achieved in programming in one field accessible to the users of computers in all fields. A combination of subroutines written in this language may do the job that is done today by separate problem-oriented languages. A programming language like PL/1, or a language still to be created, may do the job more efficiently than FORTRAN. The idea is that the programs written will have a maximal possibility of application and that programmming examples given in the literature describing computer problems will be understood by all the readers interested.

8. Reference

Dipl.-Math. Gisela Schlotter, Verarbeitung nichtnumerischer Daten
Teil I. Unterprogramme (PI-17)

Deutsches Rechenzentrum, Darmstadt

19

SEGMENTING NATURAL LANGUAGE BY ARTICULATORY FEATURES

David Shillan
Cambridge Language Research Unit,
ENGLAND.

1. For many purposes it is necessary to segment text into units convenient for handling. The sentence has been generally accepted as the natural unit, since there was no obvious alternative other than the word - which by itself tells us too little - or the paragraph - which is a vague and shifting unit, unless redefined. But the sentence is not satisfactory either: it is very variable in length; studies of speech show that in its conventional form it is not always recognizably present[1]; it may depend semantically upon its context up to at least paragraph length; and in any case what constitutes a sentence is not consistently defined (Fries[2] indicates more than 200 definitions).

2. There is another way of segmenting text, which does not suffer from these limitations, being based upon the rhythmical features of articulated speech. This use of the term "articulated" results from a view of language as basically speech, that is as skilled bodily movement. We have found it possible to bridge the gap between spoken language and written language by using features which both the writer and the reader of language tend to adopt from speech.

3. Studies of spoken language, particularly in relation to foreign language teaching, show agreement on at least the terminal boundary of the "tone group" which Crystal & Quirk[3] call "the most striking prosodic unit in English speech", and on which they have found experimentally a high rate of agreement by informants. Many different teaching books[4] exemplify this agreed feature, despite the lack of satisfactory instrumental evidence on continuous speech (into which research is now being planned).

4. Less agreement is found on the configuration of the whole unit which terminates in the "nucleus". Some authors refer to "tone groups" or "tone units", some to "sense groups", some use both terms: this overlapping category of tone and sense suggested a field for further study, which has been proceeding at C.L.R.U. for some time. Syntax is not usually brought into the treatment of this subject, since the approach is phonological; but among the authors

---

* Work supported by Canadian National Research Council.

referred to[4], MacCarthy does indicate that syntactic criteria determine the structure of his "intonation groups". Our studies support the work of those who suggest that what is commonly called "stress" has a semantic function[5], and what can be analysed in terms of intonation is the syntactic feature[6], - a kind of audible syntactic braketting.

5. It is common practice in the teaching of English as a foreign language (see Baird[7]) to use tone groups of two stresses (head and nucleus) as examples, but this configuration is not usually formalized. In my own use of such drill material for the foreign learner, I have for many years adopted this unit, marked it with a musical phrase-mark, and called it, since my 1954 publication[8], a "phrasing". My drill use of this unit gives a minimal context of not less than one sentence - a sentence being segmentable into one or more phrasings, the phrasing being thus a unit between the word and the sentence but not necessarily coterminous with the clause or grammatical phrase. (The musical analogy shows phrasing as a category distinct from the note, the bar, and the section.)

6. Ten years after publication of these drills, my work was called upon by Margaret Masterman[9] in relation to her own semantic approach, for which the two stress-points of the phrasing were seen to correspond to two information points. In the meantime I had been led by teaching experience to consider the difficulty of foreign learners with adequate vocabulary and adequate syntax but no adequate speech-experience of English. They were unable to read a piece of current English (e.g. a "Times" leading article) with understanding, whereas the native English reader, even if momentarily puzzled by perhaps a hastily-worded sentence, would immediately feed back into his reading of it (i.e. "in his mind's ear") the natural speech form (i.e. the phrasing) with which the writer had written it.

7. From this the conception of "stress-point" became differentiated from precise syllabic location of stress (which is itself a complex of amplitude, frequency, and duration) and was defined as the word or words centred, in stress-and-tone prominence, on the nuclear tone, and the word or words centred (in the same sense of "prominence") on that head tone which predominates above any other head or heads which might follow the preceding nucleus.

This method of dealing with tone groups which apparently have more than one head proves to be operationally satisfactory. It gives us a consistent phrasing of two beats, the second of which consists, in certain cases, of a "silent stress" (a phenomenon vouched for by many phoneticians[10]). It also helps to meet the difficulty of differently timed languages, referred to in para. 13 below.

8. It follows from the treatment of stress-points indicated in para. 7 above, that spread stress will occur in regular compounds, such as "semi+readiness", and it also occurs very frequently in cases of a noun with its qualifier, whether true adjective or noun acting as adjective, e.g. "political+requirements", or "staff+planning", and in general where we find intimately associated words on which the stress falls with virtually equal emphasis.

9. The silent beat may or may not be a perceptible pause, but tends to occur in certain typical locations, e.g. where some expression of significant semantic content is about to follow. It would also be possible in many cases to imagine the phrasing re-written using relevant syllables instead of the silent beat, e.g. "in a review of progress" instead of

"in a review ()".

In marking phrasings on text two symbols are used in addition to the + sign for spread stress and the () sign for silent beat. They are the well-known tonetic marker ` (originally representing a high falling tone) used for the nuclear stress, and the stress-mark ' used for the head stress. These may also be referred to as primary and secondary stress-points, the nucleus being primary because in general it indicates the <u>topic</u> of the utterance and the head being secondary because in general it indicates the <u>comment</u>. Thus reading down all the nuclear stress-points of a text printed as a series of phrasings one below the other, we have an index of the topic of the whole text.

10. A piece of text reading
"Politically Canada is divided into ten provinces and two territories" can be phrased-up either as

"Politically () 'Canada is ˋdivided into 'ten ˋprovinces and 'two ˋterritories" or as

ˋPolitically ()
'Canada is ˋdivided
into 'ten ˋprovinces
and 'two ˋterritories.

The "quatrain" form into which this falls proves to be very frequent, particularly at the beginning of a passage. This passage continues in two more quatrains:

'Each+province is ˋsovereign
in its 'own ˋsphere
and 'administers its ˋown
'natural ˋresources,

and upon 'such ˋresources
as 'related to ˋtopography,
'position and ˋclimate
is 'based the ˋeconomy+of+the+province.

A straightforward text of this kind offers if not a word-for-word, at least something like a phrasing-for-phrasing possibility in translation. But the translation correspondence, for French for example, is often not direct but expanded (e.g. 2 or more French for 1 English), or transposed in order. Apart from these considerations, there are many cases in which the phrasing structure resolves syntactic or semantic uncertainty. Here is a case where the lack of such a means of segmentation led to a serious mistranslation:

It 'may be ˋassumed
that an 'international ˋforce
on a 'standby ˋbasis
will 'take+shape as a ˋdevelopment
out of 'practice which has already ˋbegun.

The published translation has turned the last two lines into

"prendra une forme assez singulière, ce qu'elle a déjà commencé à faire".

11. Passages of text in various styles and of various lengths have been analysed by hand, and show a consistent tendency for this rhythm to be found. There may be physiological reasons for this. Neurological studies* show persistence of tone and rhythm in cases where normal articulation is impaired[11]. Good reasons for this rhythm to be binary include the fact that the

---

*For neurological literature I am indebted to Dr. Violet MacDermot.

rhythm of the mother's heart-beat is present even to the unborn child, and the in/out rhythm of respiration and the left/right rhythm of walking are basic to human life in general. Studies in articulatory phonetics[12] support the belief that some form of kinaesthetic activity is involved in silent reading, as well as in listening to live speech, which is why we can legitimately refer to "the rhythm of the prose" in spite of the lack, up to the present, of acoustic instrumental documentation of this.

12. Though intonation supplies the contour on which the phrasing is founded, the rhythm of stress is the more essential factor. As Tibbitts[13] says: "The correct basic stressing is *mandatory* while the intonation is *variable* within as yet undefined limits". This is the reason why the phrasing hypothesis is unaffected by differences of dialect or accent. The question of isochromicity in English prose has a literature stretching back to Joshua Steele in 1775, through Coventry Patmore in 1856, and on to its thorough experimental (though not instrumental) examination by André Classe[14] in 1939 and discussion by Abercrombie in 1951[15]. There is evidence for at least a strong tendency towards a normal regular periodicity of stress-points. Our observations suggest that a speaker tends to select and order his words so as to distribute them about these pulsations of stress in such a way that points of emphasis fall naturally upon them.

13. The question of whether the phrasing can be equally well observed in languages other than English is not included in the present paper, except by the observation that when parallel texts in English and French are analysed in this way, the French equivalent of the English phrasing, as clearly delimited by the French nuclear tone (and notwithstanding the difference between a syllable-timed and a stress-timed language[16]) supplies a form of "translation unit"[17] with a measurable rate of correspondence with the English[18].

13. Examination of given phrasings in a text of 377 phrasings, followed by another of over 900 phrasings, led Dolby[19] to say: "Phrasing length, as measured by the number of syllables, appears to be a reasonably behaved statistic when viewed in isolation with routine statistical tools". (See Appendix I)

14. A method of observing the phonological configuration of phrasings is to turn written text into spoken prose on magnetic tape, pass this through a suitable pitch detector and intensity detector (such as that of

the University of Grenoble or the University of Copenhagen), and record the result on mingograph scrolls. Research now being started at C.L.R.U. is comparing the output of these two sets of apparatus with that of apparatus developed in England, with a view to finding the best selection of acoustic data by which to observe the terminal point of the phrasing (frequently a steep fall or rise in pitch), and the two stress-points as peaks of frequency-plus-amplitude-plus-duration.

15. An extension of the usefulness of this unit of segmentation can be seen in algorithmic production by computer of a form of phrasing, based on observation of the criteria used in making articulatory phrasings. This has been done at C.L.R.U. by J.B. Dobson[20] in a form which while not in every single case identical with hand-marked phrasings nevertheless provides a new and operational segmentation of continuous text. As part of the work done under contract to the National Research Council of Canada, this programme is now being applied to the phrasing of a text of 20,000 words from the Canada Year Book of 1962.

16. The normal rhythmical stress can also be provided algorithmically. This makes possible a computerized ordering of the phrasings of a text alphabetically according to four different valuations, i.e.

(i) the primary (nuclear) stress;
(ii) the secondary (=head) stress;
(iii) pendants (= unstressed strings attached) to primary stress;
(iv) pendants (= unstressed strings attached) to secondary stress.

This gives a semantic concordance (called SEMCO) from which statistical and other information can be derived. The computer can process text in this way as it could not do using the sentence as a unit, and both more economically and with more information than it could by merely cutting the text into lines of the length of the computer print-out.

17. The patterning of stressed and unstressed words, i.e. of stress-points and unstressed words can be expressed as a calculus of ordered pairs, on which research is proceeding.

REFERENCES

1. C.C. Fries: "The Structure of English"; Harcourt Brace, 1952, Longmans Green, 1957.
   R. Quirk, A. Duckworth, J. Svartvik, J.P.L. Rusiecki, A.V.T. Colin: "Studies in the correspondence of prosodic to grammatical features in English"; IXth International Congress of Linguistics 1962.

2. See 1.

3. D. Crystal & R. Quirk: "Systems of prosodic and paralinguistic features in English"; Mouton, 1964.

4. Armstrong & Ward: "Handbook of English intonation"; Heffer, 1931.
   W. Stannard Allen: "Living English Speech"; Longmans Green, 1954.
   O'Connor & Arnold: "Intonation of colloquial English"; Longmans Green, 1961.
   Arnold & Gimson: "English Pronunciation Practice", London Univ. Press, 1965.
   J.T. Pring: "Colloquial English Pronunciation", Longmans Green, 1959.
   R.A. Close: "Patterns of Spoken English"; Kenyusha (Tokyo), 1954.
   R. Kingdon: "The Groundwork of English Intonation"; Lohgmans Green, 1958.
   Lado & Fries: "English Pronunciation"; Ann Arbor, 1954.
   L.A. Hill: "Stress and Intonation step by step"; Oxford, 1965.
   W.R. Lee:"An English intonation reader"; Macmillan, 1963.
   P. MacCarthy: "Endlish Pronunciation", Heffer, 1944/50.
   D. Shillan: "Spoken English", Longmans Green, 1954/65.

5. e.g. R. Gunter: in Journal of Linguistics 2, 2 Oct. 1966.

6. M.A.K. Halliday: "Some aspects of the thematic organisation of the English clause"; Rand Memorandum, Jan. 1967.

7. A. Baird: "Transformation and sequence in pronunciation", English Language Teaching XX, 2, Jan. 1966.

8. See 4.

9. Margaret Masterman: "Commentary on the Guberina hypothesis"; Methodos 57-58, XV, 1963

10. e.g. D. Jones: "Outline of English Phonetics"; Cambridge, 1932.
    D. Abercrombie: "Studies in Phonetics and Linguistics", Oxford, 1965.

11. e.g. T. Alajouanine: "Verbal realization in aphasia"; Brain 79, part I, March 1965.

12. R.H. Stetson: "Motor Phonetics"; Amsterdam, 1951.

13. E.L. Tibbitts: in English Language Teaching XXI, 1, Oct. 1966.

14. A. Classe: "The Rhythm of English Prose"; Blackwell, 1939.

15. See 10.

16. K.L. Pike: "The Intonation of American English"; Ann Arbor, 1946.

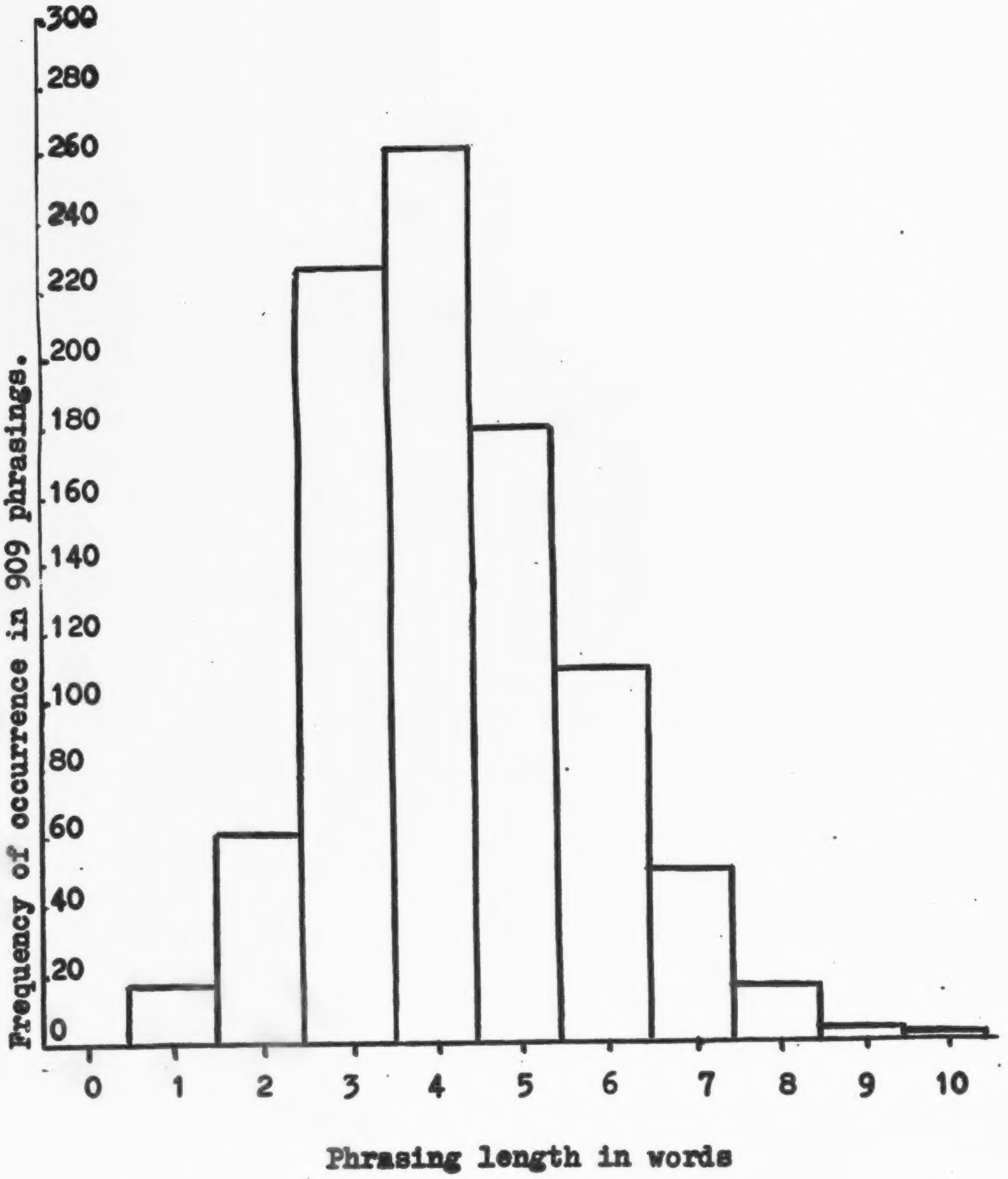
17. D. Shillan: in Meta (Montreal) XI, 3, Sept. 1966.
    D. Shillan: in English Language Teaching XXI, 2, Jan. 1967.

18. J.F. Dolby: Reports to C.L.R.U. 1965-66.

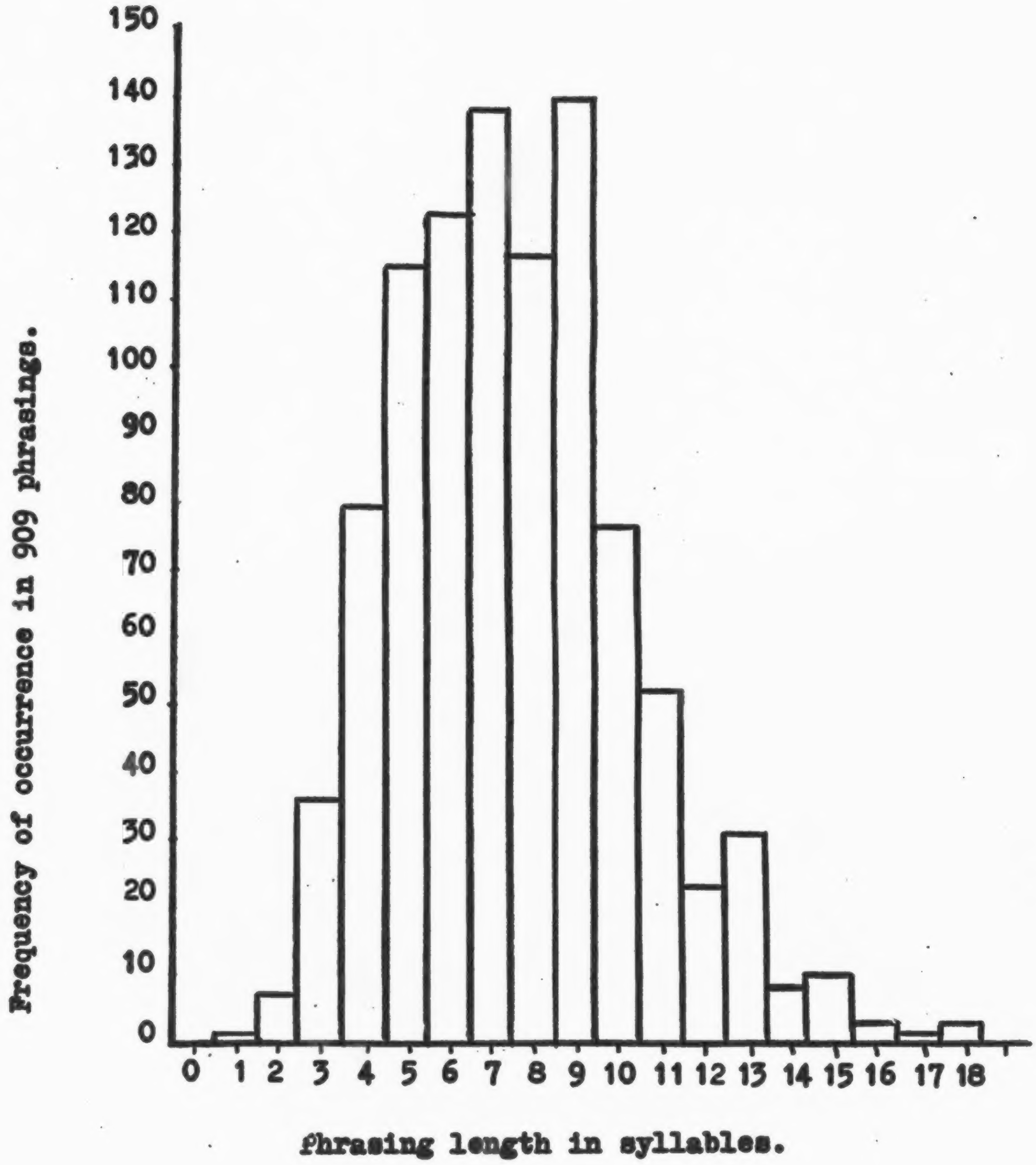
19. See 18.

20. J.E. Dobson: C.L.R.U. work paper ML 185, and later developments. (see Appendix II).

APPENDIX IA: Histogram of phrasing frequency versus phrasing length in words.



APPENDIX IB: Histogram of phrasing frequency versus phrasing length in syllables.



APPENDIX II: Computer output from phrasing program.

```
    WHILE THEY ARE WELL KNOWN AND ESTABLISHED.
I THOUGHT IT WOULD BE APPROPIATE
TO DRAW YOUR ATTENTION
TO CERTAIN OF THE DEPARTMENTAL PROGRAMMES
THAT ARE LESS WELL KNOWN
IN RELATIONSHIP TO SERVICES
FOR THE AGED.
BUT WHICH NEVERTHLESS
CAN CONTRIBUTE SIGNIFICANTLY
TO THEIR WELL BEING.


    ONE OF THESE IS
THE NATIONAL WELFARE GRANT PROGRAMME
WHICH WAS ESTABLISHED
AS LATE AS NINETEEN SIXTYTWO
WITH CONSIDERABLE SUPPORT AND ENTHUSIASM
FROM THE PROVINCIAL GOVERNMENTS.
AND FROM THE NATIONAL AND LOCAL VOLUNTARY WELFARE
                                         AGENCIES.

    ONE MILLION DOLLARS
IS AVAILABLE
UNDER THIS PROGRAMME
DURING THE CURRENT FISCAL YEAR
AND THAT AMOUNT IS TO INCREASE
AT THE RATE OF HALF A MILLION DOLLARS
A YEAR
```

20

UN MÉTALANGAGE DE GRAMMAIRES TRANSFORMATIONNELLES

APPLICATIONS AUX PROBLEMES DE TRANSFERTS ET DE GENERATION SYNTAXIQUES

G. 2300 - A

Janvier 1967

G. VEILLON
J. VEYRUNES
B. VAUQUOIS

Cette publication présente une partie des résultats acquis par l'équipe de recherche sur les modèles au cours de l'année 1966. Cette partie est essentiellement consacrée à la description de l'outil métalinguistique utilisé pour les grammaires transformationnelles du système de traduction automatique. Ces grammaires feront l'objet de publication séparées lorsqu'elles seront suffisamment au point.

Les auteurs expriment leur gratitude à Monsieur le Professeur J. P. BENZECRI qui a bien voulu s'intéresser à cette étude et à la D.R.M.E. dont la participation financière dans le cadre du contrat 65-34-497 en a permis la réalisation.

PLAN

- INTRODUCTION

I - TRANSFORMATION DE STRUCTURES ARBORESCENTES
  - 1° - Définition des structures arborescentes
    - a) n-graphe
    - b) Structure
    - c) Structure arborescente
  - 2° - Représentation des structures arborescentes et des arborescences munies d'une relation d'ordre restreinte
  - 3° - Chemins et relations sur une structure arborescente
    - a) Propriétés possédées par un sommet
    - b) Construction des relations simples
    - c) Construction des relations multiples
  - 4° - Reconnaissance d'une figure dans une structure arborescente
    - a) Schéma de figure
    - b) Figures dans une arborescence munie d'une relation d'ordre restreinte
    - c) Figures dans une structure arborescente
  - 5° - Transformations
    - a) Famille d'arborescences
    - b) Décomposition d'une arborescence
    - c) Insertion d'arborescence
    - d) Transformations synthétiques
    - e) Transformations
  - 6° - Grammaires transformationnelles
    - a) Règle de transformation sur une structure
      - α - partie gauche
      - β - partie droite
      - γ - application d'une règle de transformation
      - δ - écriture et exemples
    - b) Grammaires transformationnelles

II - PROGRAMMATION

- 1° - Représentation d'une structure
- 2° - Description de la reconnaissance
  - a) Balayage général
  - b) Recherche d'une figure correspondant à une partie gauche de règle donnée
- 3° - Application des transformations

III - APPLICATION AUX PROBLEMES DE TRADUCTION AUTOMATIQUE

- 1° - Le rôle des grammaires transformationnelles dans le système de traduction automatique
- 2° - Application au modèle M3

INTRODUCTION

Au cours de ces dernières années les grammaires transformationnelles ont occupé une place privilégiée dans la littérature relative aux systèmes de structures linguistiques, aux problèmes de génération syntaxique et aussi, éventuellement, à quelques problèmes de reconnaissance liés à la traduction automatique [1], [2], [3], [4], [5]. Ces grammaires transformationnelles et, plus généralement, les transformations brillent par le nombre de définitions dont elles ont été l'objet. En effet, un auteur qui étudie un problème linguistique adapte la notion de transformation dont il a besoin au but qu'il poursuit. Attachée à un système linguistique cette notion revet une signification particularisée par le problème traité. La transformation indique comment, quelquefois explique pourquoi tels phénomènes se présentent. A propos des problèmes que nous avons rencontrés lors de la réalisation de certains modèles du système de traduction automatique nous avons également été conduits à définir des transformations. A cette multiplicité des définitions s'oppose, en général, une certaine faiblesse des moyens d'expression et des algorithmes d'utilisation. En effet, il est souvent proposé des "règles de transformation" mais le formalisme de leur écriture ou bien n'est pas rigoureusement défini ou bien se heurte à des difficultés matérielles insurmontables si l'on veut établir toutes les règles d'un modèle réel et non d'un modèle réduit. En outre, des problèmes liés aux algorithmes d'emploi de telles grammaires transformationnelles sont peu souvent abordés. Sans prétendre résoudre tous ces problèmes, à partir des notions de transformations déjà rencontrées et de celles que notre système de traduction automatique faisait apparaitre, nous avons essayé de systématiser cette notion.

- Les "modèles transformationnels" en traduction automatique

Rappelons la place de ces modèles dans le système de traduction automatique du C.E.T.A. - L'article "syntaxe et interprétation" [6] précise le modèle de reconnaissance syntaxique des phrases de la langue source (modèle M2). Ce que l'on obtient à la sortie de ce modèle est une structure arborescente (éventuellement plusieurs) de la phrase proposée. Cette structure établit un système de liaisons entre les mots qui constituent la phrase en indiquant pour chaque liaison la règle de grammaire qui a permis de l'obtenir ainsi que l'élément considéré comme gouverneur.

Le but poursuivi par le modèle suivant (M3) est de transformer cette structure pour obtenir une évaluation sémantique des liaisons syntaxiques. On est donc conduit à substituer à une ou plusieurs règles formelles, une "étiquette" (telle que "agent", "objet", "conséquence", etc...) qui doit indiquer la fonction du mot ou groupe de mots considéré dans la phrase. En outre, pour éviter la prolifération des ambiguités de construction syntaxique, le modèle M2 peut associer à une phrase une seule structure qui, en fait, représente une famille de structures bien déterminées. En conséquence le modèle M3 doit éventuellement corriger le système de liaisons proposé par le modèle précédent pour aboutir à une articulation correcte des fonctions de chaque mot dans la phrase.

Le problème de la transformation consiste donc à remplacer une structure arborescente par une autre en changeant les noms attribués aux sommets, en supprimant ou en rajoutant certains sommets, enfin en modifiant la relation d'ordre du graphe initial.

Le modèle suivant (M'3) fait apparaître le même problème. Il s'agit maintenant de transformer le graphe associé à une phrase dans le modèle M3 (sortie de langage pivot) pour en déduire une construction syntaxique de surface dans la langue cible.

Ainsi, les transformations qui interviennent dans ces deux modèles, de même que celles qui sont mentionnées dans la littérature peuvent se ramener à un problème mathématique unique. C'est celui de l'application d'un ensemble de graphes dans un autre ensemble de graphes. Il semble d'après l'usage actuel que l'on fait des transformations, que l'on puisse limiter le problème précédent à des graphes qui sont seulement des arborescences sans perte de généralité. Encore faut-il mentionner que ces [illegible] peuvent comporter des relations d'ordre latéral, c'est-à-dire que tous les descendants directs d'un sommet quelconque sont totalement ordonnés.

Dans ce qui suit nous nous proposons d'étudier les transformations de structures arborescentes d'un point de vue formel en soulignant quelques problèmes (non résolus) et en examinant un type de langage outil pour la description de ces transformations. Enfin, la dernière partie montre l'application de ce métalangage à l'écriture des grammaires des modèles M3 et M'2.

I - TRANSFORMATIONS DE STRUCTURES ARBORESCENTES

1° - Définition des structures arborescentes

a) n-graphe

Un graphe (X,Γ) est la donnée d'un ensemble X et d'une application multivoque Γ de X dans X - On peut dire aussi que (X,Γ) est la donnée d'une partie de X^2.

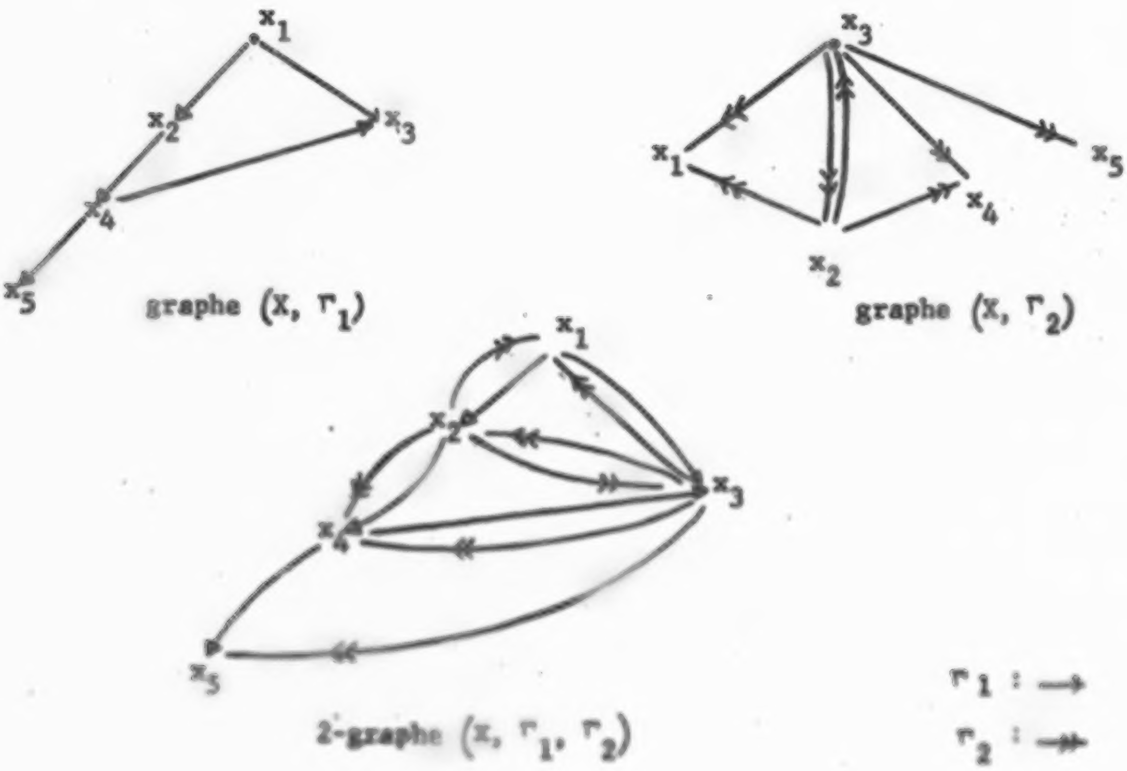
Soient Γ_1, Γ_2, ..., Γ_n des applications multivoques de X dans X. Nous appellerons n-graphe le (n+1)-uple : $(X, \Gamma_1, \Gamma_2, \dots, \Gamma_n)$ considéré comme le graphe union des graphes (X, Γ_1), (X, Γ_2), ..., (X, Γ_n) dans lequel tous les arcs sont désignés selon leur appartenance à l'application d'où ils proviennent

Exemple Soient : $X = (x_1, x_2, x_3, x_4, x_5)$

$$\Gamma_1 : \begin{cases} x_1 \longrightarrow x_2, x_3 \\ x_2 \longrightarrow x_4 \\ x_4 \longrightarrow x_3, x_5 \end{cases}$$

$$\Gamma_2 : \begin{cases} x_3 \longrightarrow x_1, x_2, x_4, x_5 \\ x_2 \longrightarrow x_1, x_3, x_4 \end{cases}$$

Le 2-graphe (X, Γ_1, Γ_2) est représenté par la figure 1



graphe (X, Γ_1)

graphe (X, Γ_2)

2-graphe (X, Γ_1, Γ_2)

figure 1

b) Structure

Soit un n-graphe $(X, \Gamma_1, \dots, \Gamma_n)$. Désignons par V un ensemble de symboles (vocabulaire ou alphabet), et soit Δ une application univoque de X dans V.

Nous appellerons "structure", le (n+2)-uple : $(X, V, \Gamma_1, \dots, \Gamma_n, \Delta)$

On obtient ainsi la structure à partir du n-graphe en attribuant à chaque sommet x_i du n-graphe le symbole appartenant à V au moyen de Δ.

Ainsi des sommets distincts de la structure peuvent être désignés par le même symbole.

Exemple Reprenons le 2-graphe du paragraphe précédent et posons :
$V = (a, b, c)$

Δ :

x_1	x_2	x_3	x_4	x_5
a	b	c	a	c

La structure $(X, V, \Gamma_1, \Gamma_2, \Delta)$ est indiquée sur la figure 2

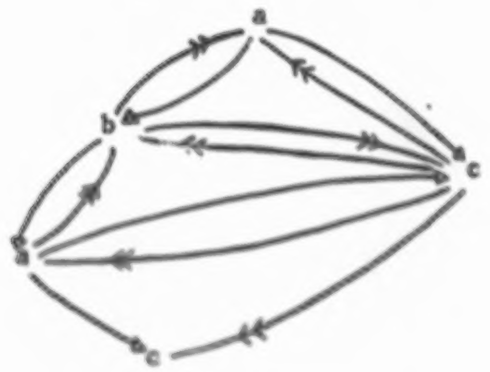


figure 2

c) Structure arborescente

On appelle "structure arborescente" une structure $\Sigma = (X, V, \Gamma_1, \Gamma_2, \Delta)$ où

- le graphe (X, Γ_1) est une arborescence, c'est-à-dire :

    1) il existe un sommet $x_o \in X$ unique tel que $\Gamma_1 x_o = \emptyset$

    2) pour tout sommet $x \neq x_o$, $\Gamma_1 x = y$, $y \in X$

    3) le graphe est connexe

- L'application Γ_1 étant déterminée, l'application Γ_2 doit satisfaire aux conditions suivantes :

Le graphe (X, Γ_2) est tel que

- à tout sous-ensemble $\Gamma_1^{-1}x$ correspond une composante connexe de (X, Γ_2)

- pour chacun de ces sous-ensembles $\Gamma_1^{-1}x$, la relation d'ordre associée à Γ_2 est totale. Ainsi, dans tout sous-ensemble $X_i = \Gamma_1^{-1}x_i$, il existe un $x_A \in X_i$ unique tel que $\Gamma_2^{-1}x_A = \emptyset$ et un $x_B \in X_i$ unique tel que $\Gamma_2\, x_B = \emptyset$

La donnée du 2-graphe (X, Γ_1, Γ_2) constitue une arborescence munie d'une relation d'ordre restreinte telle que l'a définie HOLT [7]
On peut interprêter le graphe (X, Γ_1) comme une relation de "paternité" :

"y est père de x" $\Longleftrightarrow \Gamma_1\, x = y$

De même, le graphe (X, Γ_2) exprime un ordre entre les fils d'un même père :

$\Gamma_2 x = y$ implique que x est né avant y

Un sommet de type x_A est dit "Ainé" ; un sommet de type x_B est dit "Benjamin".

Exemple Soit le 2-graphe (X, Γ_1, Γ_2) où

$X = (A, B, C, D, E, F, H, I)$

Γ_1 :	A	B	C	D	E	F	G	H	I
	$\emptyset$	A	B	B	B	A	A	G	G

Γ_2 :	A	B	C	D	E	F	G	H	I
	$\emptyset$	F	D	E	$\emptyset$	G	$\emptyset$	I	$\emptyset$

représenté par la figure 3, qui est une arborescence munie d'une relation d'ordre restreinte.

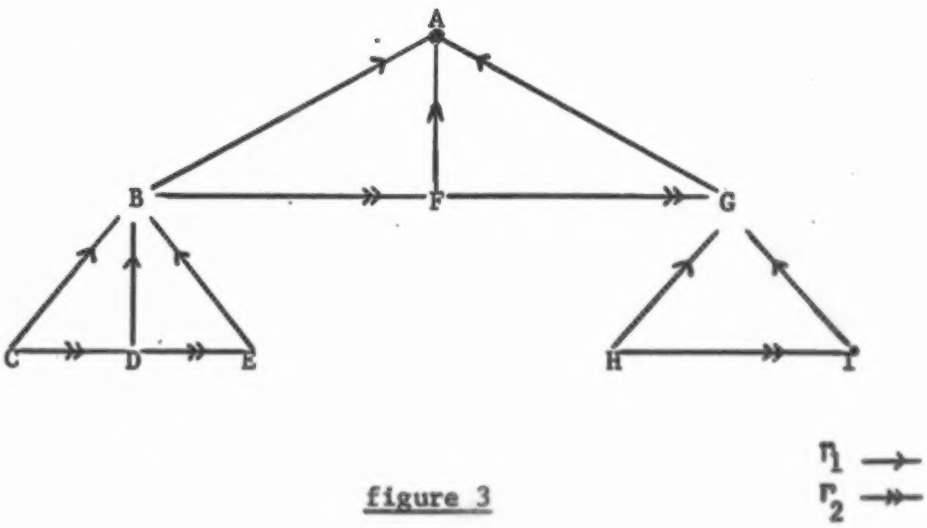

A
B
F
G
C
D
E
H
I
Γ_1
Γ_2

figure 3

Si nous nous donnons en outre :

$V = (a, b, c, d, e)$

et Δ :

A	B	C	D	E	F	G	H	I
a	b	d	e	e	c	b	d	e

La structure $\Sigma = (X, V, \Gamma_1, \Gamma_2, \Delta)$ est une structure arborescente - cf figure 4

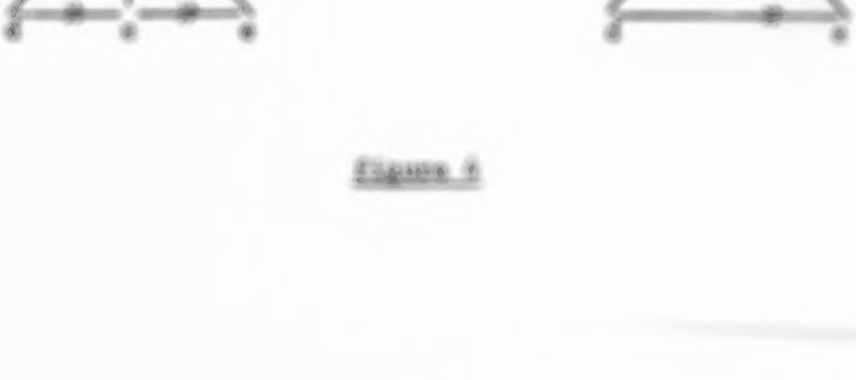


figure 4

2° - Représentation des structures arborescentes et des arborescences munies d'une relation d'ordre restreinte :

On s'attache surtout à la représentation des arborescences munies d'une relation d'ordre restreinte, celle des structures arborescentes se déduisant facilement de la première.
Pour des raisons algorithmiques, nous représenterons une telle arborescence à l'aide de trois applications biunivoques, notées α, γ et π, et définies de la façon suivante :

- $\alpha(x) = y$ si et seulement si $\Gamma_1 y = x$ et $\Gamma_2^{-1} y = \emptyset$ (fils ainé)
- $\gamma(x) = y$ si et seulement si $\Gamma_2 x = y$ (frère cadet)
- $\pi(x) = y$ si et seulement si $\Gamma_1 x = y$ et si $\Gamma_2 x = \emptyset$ (père du benjamin)

Exemple

Reprenons l'arborescence du paragraphe précédent (figure 3). Sa représentation au moyen des fonctions α, γ et π est alors celle de la figure 5 où l'on a représenté aussi la structure arborescente déjà mentionnée.

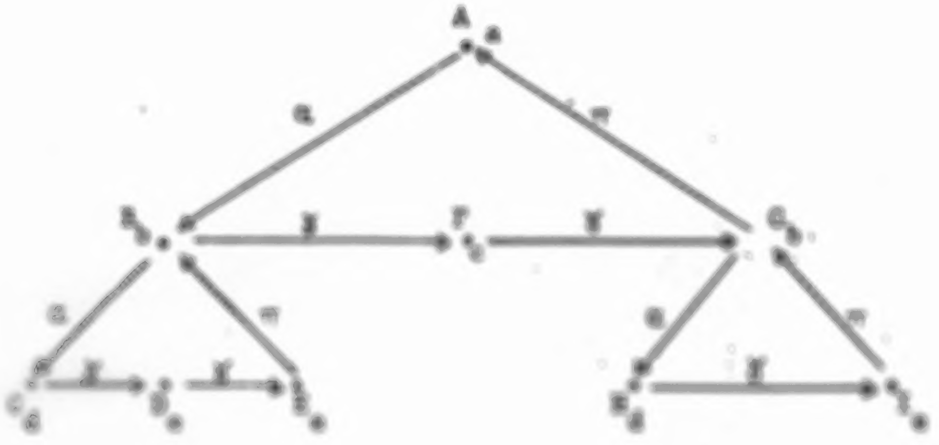


figure 5

On retrouve certaines des fonctions décrites par A. HOLT [7] pour représenter les arborescences munies d'une relation d'ordre restreinte.
On peut aussi considérer cette structure comme la donnée du 3-graphe (X, α, γ, π) du vocabulaire V et de l'application Δ $(\Delta : X \rightarrow V)$

3° - Chemins et relations sur une structure arborescente

D'après les définitions de α, γ, π à partir de Γ_1 et Γ_2, pour tout couple de sommets distincts (x, y), il existe au moins un chemin d'origine x et d'extrémité y et il existe un chemin unique sans circuit. Ces chemins correspondent à des compositions des fonctions α, γ, π.

Soit f une composition de α, γ, π et $x \in X$ un sommet - Si fx existe, c'est un sommet y unique de la structure ; en outre $f^{-1}y$ existe et est le seul sommet x.

Un tel chemin caractérise une relation entre deux sommets.

Nous nous proposons de déterminer les moyens de construire des relations qui possèdent les propriétés suivantes :

- des relations 1-aire correspondant à des propriétés possédées par un sommet considéré isolément.

- des relations binaires que nous rangeons dans l'une des deux classes suivantes :

- des "relations simples" - Ces relations sont telles qu'à tout x_i il corresponde au plus un y_i en relation avec x_i.

- des "relations multiples" - A tout x_i il correspond un ensemble $X_i \subset X$ de sommets y tels que $x_i R y$ $(y \in X_i)$, X_i possèdent en général plus d'un élément. On veut alors que la relation multiple induise un ordre total sur les éléments de X_i - L'importance de cette propriété apparaîtra lorsque l'on travaillera sur la structure où plusieurs sommets peuvent porter le même nom (symbole de V).

Toute relation peut être considérée comme un prédicat.
Si R est une relation 1-aire, R_x est un prédicat à 1 variable (x possède la propriété R).

Si R est une relation binaire, x R y est un prédicat à deux variables et l'ensemble X_i des sommets y liés à x_i par R est l'extension du prédicat $x_i Ry$.

$$X_i = \{y / x_i Ry\}$$

a) **Propriétés possédées par un sommet**

D'après les définitions de α, γ, π certains sommets possèdent des propriétés particulières. Ces propriétés sont caractérisées par un prédicat.

Exemples - "Etre la racine de l'arborescence" désigné par $\mathcal{P}$

Ainsi $\mathcal{P}x$ est un prédicat vrai si et seulement si x est la racine de l'arborescence :

$$\mathcal{P}x \leftrightarrow \gamma x = \emptyset \wedge \pi x = \emptyset$$

- "Etre un fils aîné" - $\mathcal{A}$

$$\mathcal{A}x \leftrightarrow \gamma^{-1}x = \emptyset$$

- "Etre un benjamin" - $\mathcal{B}$

$$\mathcal{B}x \leftrightarrow \gamma x = \emptyset$$

- "N'avoir pas de fils" - $\mathcal{F}$

$$\mathcal{F}x \leftrightarrow \alpha x = \emptyset$$

A partir de ces propriétés élémentaires on peut construire des propriétés plus compliquées

Par exemple

- "Etre fils unique" - $\mathcal{U}$

$$\mathcal{U}x \leftrightarrow \mathcal{A}x \wedge \mathcal{B}x$$

(On peut aussi écrire $\mathcal{U}x \leftrightarrow x = \alpha\pi x$)

- "Etre le point unique de l'arborescence" - $\mathcal{T}$

$$\mathcal{T}x \leftrightarrow \mathcal{P}x \wedge \mathcal{F}x$$

b) Construction des relations simples

Le cas le plus simple de construction de telles relations est celui qui provient d'une composition déterminée de fonctions α , γ , π décrivant un chemin sans circuit.

Exemple : Etablir la relation : "3ième FS" où

x 3ième FS y signifie "y est le troisième fils de x"

Alors :

$$x \text{ 3ième FS } y \longleftrightarrow y = \gamma^2 \alpha x$$

En nous referrant à l'arborescence de la figure 5

A 3ième FS G est vraie et G est le seul sommet qui soit dans cette relation avec A.

Un cas plus compliqué est celui où la relation x R y est satisfaite si et seulement si, f_1 , f_2, ... , f_n étant des compositions de α, γ , π , il existe au plus un j $(1 \leqslant j \leqslant n)$ tel que $y = f_j x$; le nombre n n'étant pas connu mais seulement majoré par le nombre d'éléments de X. On utilise alors des prédicats munis de quantificateurs.

Exemples : Etablir la relation : "FNE" où

x FNE y signifie "y est le frère ainé de x"

Alors :

$$x \text{ FNE } y \longleftrightarrow \bigvee_{\nu=0}^{n} y = \alpha\pi\gamma^{\nu}x$$

On voit qu'ici la relation peut être réflexive si x est lui-même un ainé. Si l'on veut éliminer ce cas, il suffit de définir FNE comme la conjonction du prédicat déjà écrit et du prédicat $y \neq x$ (ou bien $\neg\, \mathcal{A}(y)$)

- Etablir la relation "PERE" où

x PERE y signifie y est le père de x

$$x \text{ PERE } y \longleftrightarrow \bigvee_{\nu=0}^{n} y = \pi\gamma^{\nu}x$$

Comme la relation simple x AINE y (signifiant "y est le fils ainé de x") est définie par $x \text{ AINE } y \longleftrightarrow y = \alpha x$,

on peut aussi définir FNE par

$$x \text{ FNE } y \longleftrightarrow \bigvee_{z} \left[(x \text{ PERE } z) \wedge (z \text{ AINE } y)\right]$$

- Etablir la relation "FRERJ" où

x FRERJ y signifie "y est le plus jeune frère de x

$$x \text{ FRERJ } y \longleftrightarrow \bigvee_{t=0}^{\infty} (y = F^{t}x \wedge \overline{F}y)$$

A toute relation simple R, on associe la fonction R_f définie par $y = R_f x$ si et seulement si xRy

c) Construction des relations multiples

Dans de telles relations xRy peut être vérifié par un ensemble de sommets y pour un x déterminé, et

$$X_i = \left\{ y / x_i \, R \, y \right\} \quad \text{doit être totalement ordonné}$$

On donne alors le moyen d'énumérer récursivement les éléments de X_i - Dans ce but, on détermine un premier élément y_1 de la même manière qu'on a construit les relations simples - Puis on donne le moyen de calculer y_{i+1} connaissant y.

Exemples - Etablir la relation "FRERE" où

x FRERE y signifie que "y est frère plus jeune de x"

Alors :

$$\left\{ y / x \text{ FRERE } y \right\} = \begin{cases} y_1 = F x \\ y_{i+1} = F y_i \end{cases}$$

Etablir la relation "FILS" où

x FILS y signifie que "y est fils de x"

$$\left\{ y / x \text{ FILS } y \right\} = \begin{cases} y_1 = A x \\ y_{i+1} = F y_i \end{cases}$$

Etablir la relation "COFILS" où

x COFILS y est vraie si et seulement si

y = x ou y est un fils du père de x

$$\left\{ y / x \text{ COFILS } y \right\} = \begin{cases} y_1 = PER_f x \\ y_{i+1} = F y_i \end{cases}$$

L'ordre total induit est celui donné par F sur tous les fils du même père.

Enfin, établissons la relation "GOUV" où

x GOUV y est vraie si et seulement si

$y = \gamma^{-1}x$ (x est le cadet de y)

ou y se trouve dans la lignée des benjamins du premier y.

$$\{y \,/\, x \text{ GOUV } y\} = \begin{cases} y_1 = \gamma^{-1}x \\ y_{i+1} = \text{DFILS}_f y_i \end{cases}$$

où DFILS est une relation simple (y est le dernier fils de x)

$$x \text{ DFILS } y \longleftrightarrow \bigvee_{r=0}^{n} y = \gamma^r \alpha x \wedge \mathfrak{B}y$$

Ainsi une relation multiple s'exprime au moyen de deux relations simples, c'est-à-dire au moyen de 2 fonctions f et g

$$\begin{cases} y_1 = fx \\ y_{i+1} = gy_i \end{cases}$$

L'élément y_r de rang r dans l'ensemble $X_i = \{y \,/\, xRy\}$ est alors obtenu

par $y_r = g^{r-1}fx$

Si $g^{r-1}fx$ n'existe pas, c'est que X_i contient moins de r éléments.

Pour que chaque élément ne soit énuméré qu'une fois, il est nécessaire que

$$\bigwedge_{i=1}^{n} \bigwedge_{p=1}^{n} y_{i+p} \neq y_i$$

4° - Reconnaissance d'une figure dans une structure arborescente

a) Schéma de figure

Soient $R_1, R_2, \ldots, R_n$ des relations binaires (simples ou multiples). Soient $P_1, \ldots P_m$ des propriétés.

On définit un schéma de figure F comme une expression formelle construite sur le vocabulaire terminal $\{R_1, \ldots, R_n, P_1, \ldots, P_m, (,)\}$ de la manière suivante

(1) Si R_i est une relation alors (R_i) est un schéma de figure

(2) Si P_i est une propriété, alors (P_i) est un schéma de figure

(3) Si F est un schéma de figure et R_i une relation alors (R_iF) est un schéma de figure

(4) Si F_1 et F_2 sont deux schémas de figure, alors F_1F_2 est un schéma de figure

(5) Tous les schémas de figure sont donnés par (1),(2),(3) et(4).

En utilisant le symbolisme de définition d'ALGOL on a :

$$\langle R\rangle ::= R_1 \mid R_2 \mid \cdots \mid R_n$$

$$\langle P\rangle ::= P_1 \mid P_2 \mid \cdots \mid P_m$$

$$\langle F\rangle ::= \underset{(1)}{(\langle R\rangle)} \mid \underset{(2)}{(\langle P\rangle)} \mid \underset{(3)}{(\langle R\rangle\ \langle F\rangle)} \mid \underset{(4)}{\langle F\rangle\ \langle F\rangle}$$

Exemple

L'expression : $\left(R_{i_1}\left(R_{i_2}\left(P_{j_1}\right)\right)\left(R_{i_3}\right)\left(R_{i_4}\right)\right)\left(R_{i_5}\left(P_{j_2}\right)\right)$

où $1 \leqslant i_k \leqslant n$ pour $1 \leqslant k \leqslant 5$ et $1 \leqslant j_\ell \leqslant m$ pour $1 \leqslant \ell \leqslant 2$

est un schéma de figure F.

En effet

$\left(P_{j_1}\right)$ est un schéma de figure par la règle de construction (2)

$\left(R_{i_2}\left(P_{j_1}\right)\right)$ ------------------------------------ (3)

$\left(R_{i_3}\right)$ ------------------------------------ (1)

$\left(R_{i_2}\left(P_{j_1}\right)\right)\left(R_{i_3}\right)$ ------------------------------------ (4)

$\left(R_{i_4}\right)$ ------------------------------------ (1)

$\left(R_{i_2}\left(P_{j_1}\right)\right)\left(R_{i_3}\right)\left(R_{i_4}\right)$ ------------------------------------ (4)

$\left(R_{i_1}\left(R_{i_2}\left(P_{j_1}\right)\right)\left(R_{i_3}\right)\left(R_{i_4}\right)\right)$ ------------------------------------ (3)

$\left(P_{j_2}\right)$ ------------------------------------ (2)

$\left(R_{i_5}\left(P_{j_2}\right)\right)$ ------------------------------------ (3)

$\left(R_{i_1}\left(R_{i_2}\left(P_{j_1}\right)\right)\left(R_{i_3}\right)\left(R_{i_4}\right)\left(R_{i_5}\left(P_{j_2}\right)\right)\right)$ ------------------------------------ (4)

Considérons maintenant une arborescence munie d'une relation d'ordre restreinte $\mathcal{A}$.
Soient X l'ensemble des sommets de $\mathcal{A}$, x_o un sommet quelconque ($x_o \in X$) et F un schéma de figure.
On dit que x_o admet le schéma de figure F (dans $\mathcal{A}$) si et seulement si :

α) la proposition F (x_o) est vraie, F (x_o) étant la proposition déduite du prédicat F (x) défini de la manière suivante :

Soient F_1, F_2, F_3 des schémas de figure

Si F_1 est construit au moyen de la règle (1)

$$F_1 = (R_i) \quad \text{alors} \quad F_1\ (x) \longleftrightarrow \bigvee_y xR_iy \qquad x,\ y \in X$$

Si F_1 est construit au moyen de la règle (2)

$$F_1 = (P_j) \quad \text{alors} \quad F_1\ (x) \longleftrightarrow P_j\ (x) \qquad x \in X$$

Si F_2 est construit au moyen de la règle (3)

$$F_2 = (R_i\ F_1) \quad \text{alors} \quad F_2\ (x) \longleftrightarrow \bigvee_y \left[xR_iy \wedge F_1\ (y)\right] \qquad x,\ y \in X$$

Enfin si F_3 est construit au moyen de la règle (4)

$$F_3 = F_1\ F_2 \quad \text{alors} \quad F_3\ (x) \longleftrightarrow F_1\ (x) \wedge F_2\ (x) \qquad x \in X$$

Ainsi le prédicat associé au schéma de figure F donné en exemple est :

$$F(x) \longleftrightarrow \bigvee_{x_1}\left[xR_{i_1}x_1 \wedge \bigvee_{x_2}\left[x_1R_{i_2}x_2 \wedge P_{j_1}(x_2)\right] \wedge \bigvee_{x_3} x_1R_{i_3}x_3 \wedge \bigvee_{x_4} x_1R_{i_4}x_4\right] \wedge \bigvee_{x_5}\left[xR_{i_5}x_5 \wedge P_{j_2}(x_5)\right]$$

où $x,\ x_1,\ x_2,\ x_3,\ x_4,\ x_5 \in X$

β) les sommets qui interviennent en tant que variables liées dans les quantificateurs sont tous les éléments différents de X.
Pour formaliser cette deuxième condition, on remarque que chaque relation introduit un sommet ; donc si F comporte p relations, $x_1, \ldots, x_p$ doivent être tous différents les uns des autres.

Alors soit le prédicat $D_2\ (u_1, u_2) \longleftrightarrow u_1 \neq u_2$

Par récurrence on construit :

$$D_{k+1}\ (u_1, \ldots, u_k, u_{k+1}) \longleftrightarrow D_k\ (u_1, \ldots, u_k) \wedge \bigwedge_{j=1}^{k} D_2\ (u_j, u_{k+1})$$

Enfin on pose : D_1 (u) toujours vrai.

Ainsi, F étant un schéma de figure à p relations, la proposition "x_o admet le schéma de figure F (dans $\mathcal{A}$)" $\longleftrightarrow F(x_o) \wedge D_{p+1}(x_o, x_1, \ldots, x_p)$

L'extension de ce prédicat est l'ensemble des $x \in X$ qui rendent ce prédicat vrai.

b) Figures dans une arborescence munie d'une relation d'ordre restreinte

Désignons toujours par $\mathcal{A}$ cette arborescence ; X est l'ensemble de ses sommets et $x_o \in X$ et F un schéma de figure comportant n relations.

Au prédicat d'1 variable F (x) on associe le prédicat à n + 1 variables

$F^{(n+1)}(x, x_1, \ldots, x_n)$ déduit de F (x) en supprimant les quantificateurs qui lient les variables $x_1, \ldots, x_n$.

On appelle φ une figure de schéma F et de pivot x_o dans $\mathcal{A}$ tout élément de l'ensemble des (n+1)-uples $(x_o, x_1, \ldots, x_n)$ où $x_o, x_1, \ldots, x_n \in X$ et $F^{(n+1)}(x_o, x_1, \ldots, x_n)$ est vrai.

On peut alors définir un ordre total sur les figures de schéma F et de pivot x_o de la manière suivante

Soient $\varphi = (x_o, x_1, \ldots, x_n)$ et $\varphi' = (x_o, x'_1, \ldots, x'_n)$ deux telles figures

Soit k le plus petit indice tel que $x_k \neq x'_k$ ($k \neq 0$ puisque x_o est pivot, et $k \leqslant n$ puisque φ et φ' sont différentes)

A x_k et x'_k est associée la relation R_{i_k} telle que il existe un $j<k$ avec

$$x_j = x'_j \quad \text{et} \quad x_j \, R_{i_k} \, x_k \text{ dans } \varphi \text{ et } x_j \, R_{i_k} \, x'_k \text{ dans } \varphi'.$$

La relation R_{i_k} est évidemment une relation multiple (sinon $x_k = x'_k$) ; on peut alors écrire x_k et x'_k sous la forme

$$x_k = g^r f \, x_j \quad \text{et} \quad x'_k = g^{r'} f \, x_j$$

On ordonne φ et φ' par la relation $\varphi < \varphi'$ si et seulement si $r<r'$

Montrons que cette relation sur les figures est un ordre total :

Il est facile de voir que cette relation est non réflexive (tous les x_k sont égaux) et antisymétrique (elle est déduite de la relation d'ordre sur r).

Montrons qu'elle est transitive :

En effet, si l'on a $\varphi < \varphi'$ et $\varphi' < \varphi''$, alors

soient k et ℓ les plus petits indices tels que :

$$x_k \neq x'_k \quad \text{et} \quad x'_\ell \neq x''_\ell$$

si min $(k, \ell) = k$, alors k est le plus petit indice tel que $x_k \neq x''_k$, car $x'_k = x''_k$, et l'on a $\varphi < \varphi''$ pour la même raison que $\varphi < \varphi'$

si min $(k, \ell) = \ell$, alors $x_\ell \neq x''_\ell$ car $x_\ell = x'_\ell$ et $\varphi < \varphi''$.

Il est bien évident que cet ordre est total, puisqu'il est défini pour tout couple de figures.

Exemple

Reprenons le schéma de figure F proposé au paragraphe 4 a dans lequel R_{i_1} est la relation FILS

R_{i_2} -------------- FILS

R_{i_3} -------------- FRERE

R_{i_4} -------------- COUSING

R_{i_5} -------------- CADET

P_{j_1} est la propriété "n'avoir pas de fils" $\mathcal{C}$ (paragraphe 3 a)

P_{j_2} est la propriété "être benjamin" $\mathcal{B}$ (paragraphe 3 a) x COUSING y étant vraie si et seulement si y est un cousin à gauche de x et définie par :

$$y_1 = \alpha\gamma^{-1}\ \mathrm{PERE}_f\ x$$

$$y_{i+1} = \begin{cases} \gamma y_i & \text{si } \gamma \neq \emptyset \\ \alpha\gamma^{-1}\pi y_i & \text{si } \gamma = \emptyset \end{cases}$$

Alors le schéma de figure s'écrit :

F : (FILS(FILS($\mathcal{C}$)) (FRERE) (COUSING))(CADET($\mathcal{B}$)))

Soit l'arborescence $\mathcal{A}$ suivante :

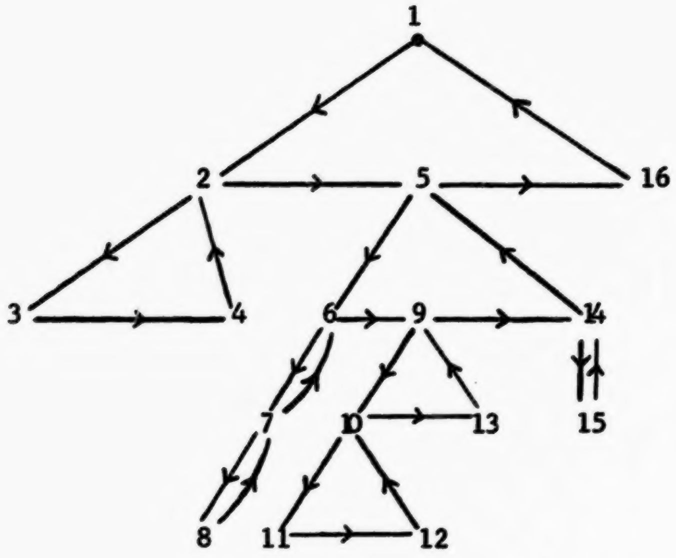


figure 6

Si l'on se donne comme pivot le sommet 5, nous obtiendrons dans l'ordre les figures suivantes :

(5, 9, 13, 14, 3, 16)
(5, 9, 13, 14, 4, 16)

Il peut se faire qu'un schéma de figure en contienne un autre.
Soit par exemple

F' : (FILS(FILS($\mathcal{C}$))(FILS(COUSING))(CADET($\mathcal{B}$)))

Du même pivot, le sommet 5, on obtient, dans l'ordre, les figures

(5, 9, 13, 6, 3, 16)
(5, 9, 13, 6, 4, 16)
(5, 9, 13, 14, 3, 16)
(5, 9, 13, 14, 4, 16)
(5, 14, 15, 6, 3, 16)
(5, 14, 15, 6, 4, 16)
(5, 14, 15, 9, 3, 16)
(5, 14, 15, 9, 4, 16)

qui contiennent les 2 figures précédentes

Remarque : Etant donné un sous-ensemble de n + 1 points quelconque d'une arborescence, $(o \leq n \leq \mathrm{card}\,(X))$, ce sous-ensemble est une figure, car on peut toujours trouver un schéma de figure correspondant.

c) Figures dans une structure arborescente

Soit S une structure arborescente ; S est donnée par une arborescence munie d'une relation d'ordre restreinte $\mathcal{A}$, et par un vocabulaire V et une application Δ de X dans V. $S = (\mathcal{A}, V, \Delta)$.

F étant un schéma de figure, à chaque relation R_{i_k} de F on associe un symbole $a_{i_k} \in V \cup \{*\}$

On associe également un symbole $a_o \in V \cup \{*\}$ à la place du pivot.

Soit $x_o \in X$ un pivot, on appelle figure de schéma F et de pivot x_o dans S, toute figure de schéma F et de pivot x_o dans $\mathcal{A}$ telle que pour tout k $o \leq k \leq n$ $\Delta x_{i_k} = a_{i_k}$ ou $a_{i_k} = *$.

Exemple

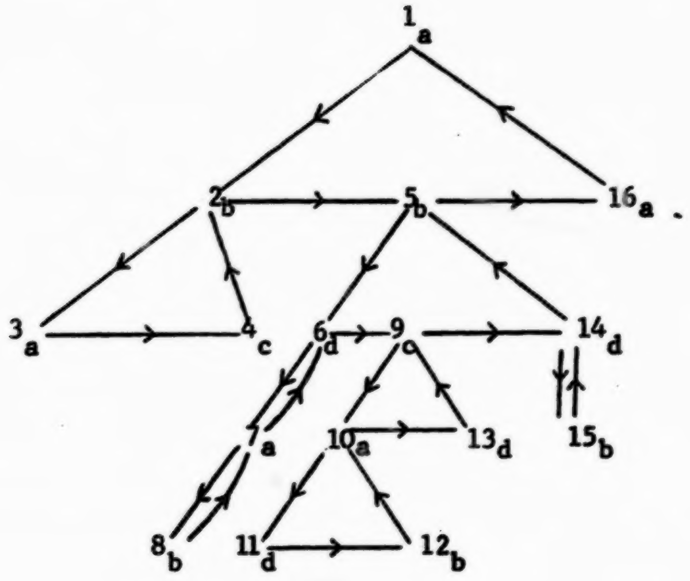


figure 7

Le vocabulaire $V = \{a, b, c, d\}$ et l'application Δ est représentée sur la figure.
Considérons le schéma de figure F' (paragraphe 4 b) qui donnait 8 figures de pivot x_5 dans $\mathcal{A}$
Alors l'expression :

$$* \left(\text{FILS } c\left(\text{FILS } *(\mathcal{C})\right)\right)\left(\text{FILS } d\left(\text{COUSING } c\right)\right)\left(\text{CADET } a\,(\mathcal{B})\right)$$

appliquée au pivot x_5 fournit dans S les deux figures

(5, 9, 13, 6, 4, 16)
(5, 9, 13, 14, 4, 16)

5° - Transformations

a) Familles d'arborescences

Soit un ensemble X et deux sous ensemble X', X" tels que $X' \subset X, X'' \subset X$ et $X' \cap X'' = \emptyset$. Soient deux arborescences $\mathcal{A}' = (X', \Gamma'_1, \Gamma'_2)$ et $\mathcal{A}'' = (X'', \Gamma''_1, \Gamma''_2)$

Nous appellerons famille d'arborescences l'ensemble $\{\mathcal{A}', \mathcal{A}''\}$ que nous pouvons noter : $(X' \cup X'', \Gamma_1, \Gamma_2$ avec $\Gamma_1 x = \Gamma'_1 x$ si $x \in X'$

et $\Gamma_1 x = \Gamma''_2 x$ si $x \in X''$

b) Décomposition d'une arborescence

Soient une arborescence (X, Γ_1, Γ_2) et un sommet x, on se propose de définir les deux applications suivantes :

Si $y \neq x$ $\quad \Gamma'_1 y = \Gamma_1 y$

$y = x$ $\quad \Gamma' y = \emptyset$

$$\left.\begin{array}{l} y \neq x \\ y \neq \Gamma_2^{-1} x \end{array}\right\} \quad \Gamma'_2 y = \Gamma_2 y$$

$y = x$ $\quad \Gamma'_2 y = \emptyset$

$y = \Gamma_2^{-1} x$ $\quad \Gamma'_2 y = \Gamma_2 x$

si $y = \Gamma_2^{-1} x$

alors $(X, \Gamma'_1, \Gamma'_2)$ est une famille de deux arborescences $\mathcal{A}', \mathcal{A}''$, telles que $X' \cup X'' = X$; Γ'_1 et Γ'_2 répondent aux axiomes de définition des arborescences, le sommet x étant tel que $\Gamma'_1 x = \Gamma'_2 x = \emptyset$ (racine)

Nous appellerons détachement de l'arborescence de racine x (DETA (x)) une telle opération. Elle a pour effet de décomposer une arborescence en deux arborescences.

En utilisant les fonctions α, γ et π, DETA (x) peut se décrire ainsi :

si $\alpha(y) \neq x$, $\alpha'(y) = \alpha(y)$

si $\alpha(y) = x$, $\alpha'(y) = \gamma(x)$

si $\gamma(y) = x$ et si $\gamma(x) \neq \emptyset$, $\gamma'(y) = \gamma(x)$

si $\gamma(x) = \emptyset$, $\gamma'(y) = \emptyset$, $\pi'(y) = \pi(x)$

si $\gamma(y) \neq x$, $y \neq x$ $\quad \gamma'(y) = \gamma(y)$

si $y = x$ $\quad \gamma'(x) = \emptyset$

si $y \neq x$, $\pi'(y) = \pi(y)$, si $y = x$, $\pi'(x) = \emptyset$

Exemple :

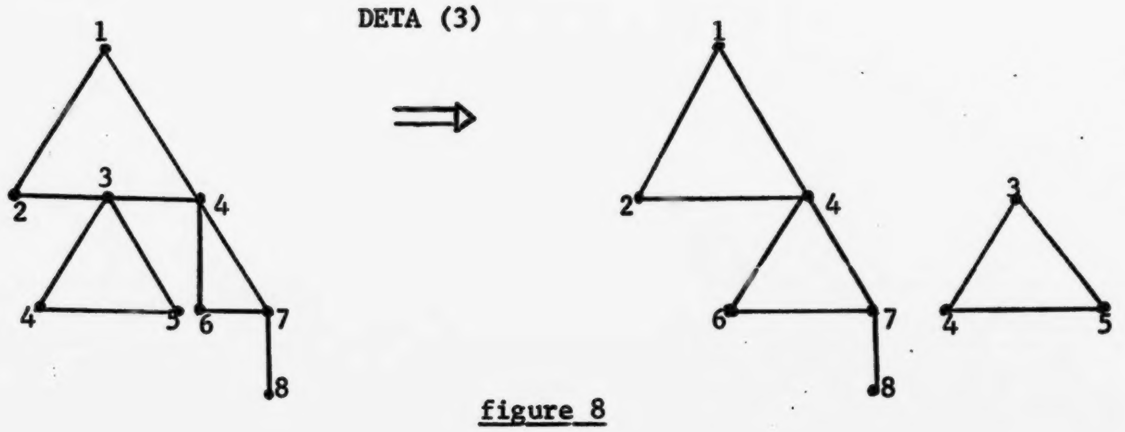


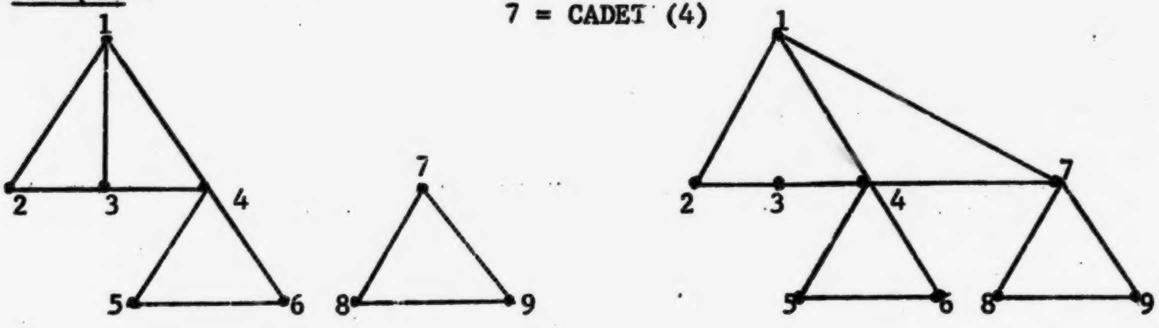
figure 8

c) Insertion d'arborescence

Soit une famille de 2 arborescences $\{\mathcal{A}', \mathcal{A}''\}$ de racines x' et x''. On se propose de construire une arborescence unique en insérant le sommet x'', racine de $\mathcal{A}''$, comme sommet de $\mathcal{A}'$. Pour cela, on définit l'insertion par une fonction f et un sommet z de X'. L'arborescence résultante sera telle que x'' = f (z), et que toutes les relations d'ordre partiel associées à Γ_1 et Γ_2 soient conservées. La fonction f doit permettre de placer x'' dans $\mathcal{A}'$. x'' étant tel que $\gamma(x'') = \pi(x'') = \emptyset$, f permet de définir $\gamma(x'')$ ou $\pi(x'')$, sans altérer $\alpha(x'')$.

On se resteindra donc à des insertions de x'' comme sommet terminal de $\mathcal{A}'$ ce qui exclut, en particulier, la fonction PERE ; le positionnement de x'' = PERE_f (z) implique de définir, en outre, la relation d'ordre entre z et les FILS de x''.

Exemple :



$\gamma_1, \alpha_1, \pi_1$

$\gamma_2(z) = x''$

$\gamma_2(x'') = \gamma_1(z)$ si $\gamma_1(z) \neq \emptyset$

$\pi_2(x'') = \pi_1(z)$ si $\gamma_1(z) = \emptyset$

figure 9

d) Transformations synthétiques

Aux transformations élémentaires de détachement et d'insertion sont adjointes des transformations synthétiques ne concernant qu'une seule arborescence.

Les transformations synthétiques définies actuellement sont au nombre de deux :

L'échange de deux sommets, ou de deux sous arborescences, noté x_1 ECH x_2 correspond aux fonctions α', γ', π' déduites ainsi de α, γ, π

- Si $\alpha(x) \neq x_1, x_2$, $\alpha'(x) = \alpha(x)$

Si $\alpha(x) = x_1$ $\quad \alpha'(x) = x_2$

Si $\alpha(x) = x_2$ $\quad \alpha'(x) = x_1$

Si $\gamma(x) \neq x_1, x_2$ $\quad \gamma'(x) = \gamma(x)$

$\gamma(x) = x_1$ $\quad \gamma'(x) = x_2$

$\gamma(x) = x_2$ $\quad \gamma'(x) = x_1$

Si $x \neq x_1, x_2$ $\quad \pi'(x) = \pi(x)$

$\gamma'(x) = \gamma(x)$

Si $x = x_1$, $\gamma(x_1) \neq \emptyset$, $\gamma'(x_2) = \gamma(x_1)$, $\pi'(x_2) = \emptyset$; $\gamma(x_1) = \emptyset$, $\pi'(x_2) = \pi(x_1)$

Si $x = x_2$, $\gamma(x_2) \neq \emptyset$, $\gamma'(x_1) = \gamma(x_2)$, $\pi'(x_1) = \emptyset$; $\gamma(x_2) = \emptyset$, $\pi'(x_1) = \Pi(x_2)$

L'échange n'est défini que si les sommets x_1 et x_2 ne sont pas comparables selon Γ_1.

Exemples :

1)

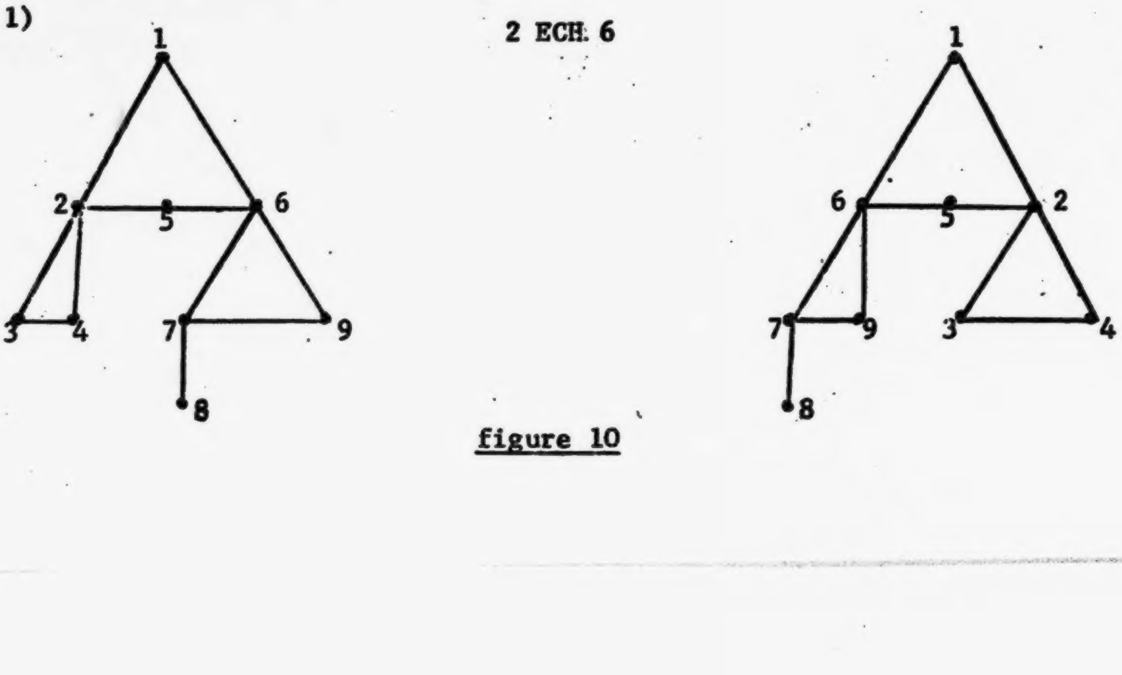


figure 10

2) Dans le cas où il existe une relation de hiérarchie (Γ_1 de l'arborescence) entre x_1 et x_2, l'opération d'échange donnera un résultat aberrant, comme le montre l'exemple suivant :

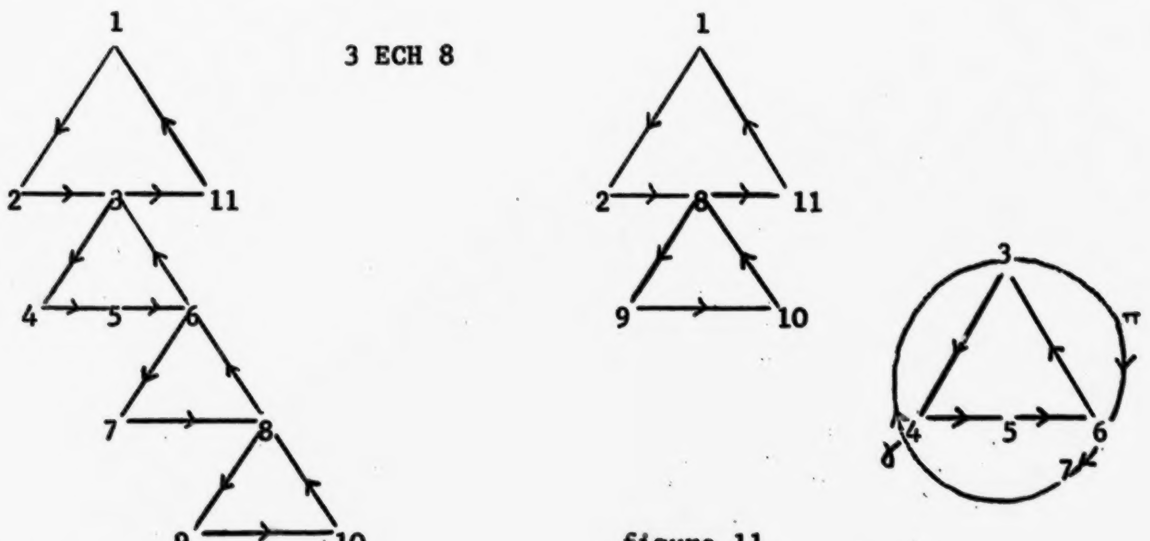


figure 11

On voit que l'arborescence n'est plus connexe. La partie correspondant au sommet de hiérarchie supérieure est séparée de l'arborescence principale. Ce n'est pas une arborescence, car il n'existe pas de racine (sommet tel que $\gamma = \pi = \emptyset$). Le résultat n'est donc ni une arborescence ni une famille d'arborescences.

Remarquons néanmoins qu'une opération de détachement sur le sommet 3 aura pour effet d'en faire la racine d'une arborescence normale.

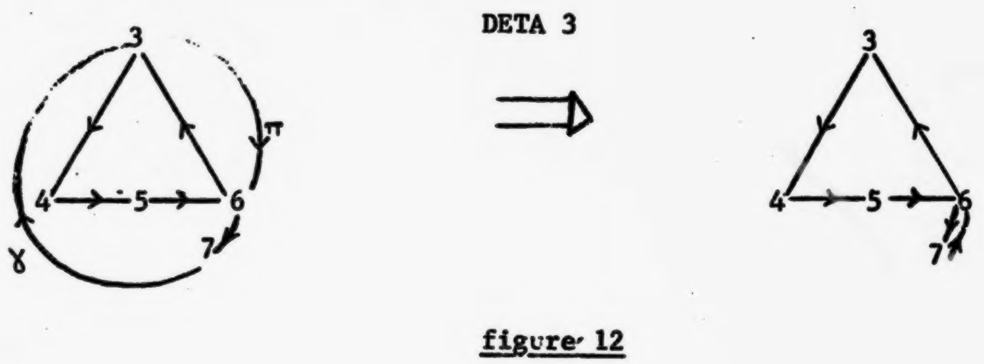


figure 12

L'autre transformation (Absorption ABS) est une suppression de sommet sans suppression de l'arborescence associée. Les relations d'ordre sur Γ_1 et Γ_2 sont préservées de la manière suivante :

soient α, γ, π et α, γ', π' les relations transformées :

ABS (x)

$$\alpha'(x) = \gamma'(x) = \pi'(x) = \emptyset$$

Si $\alpha(y) = x$, $\alpha(x) \neq \emptyset$, $\alpha'(y) = \alpha(x)$

$\alpha(x) = \emptyset$, $\gamma(x) \neq \emptyset$, $\alpha'(y) = \gamma(x)$

$\alpha(x) = \emptyset$, $\gamma(x) = \emptyset$, $\alpha'(y) = \emptyset$

$\alpha(y) \neq x$, $\alpha'(y) = \alpha(y)$

Si $\gamma(y) = x$, $\alpha(x) \neq \emptyset$, $\gamma'(y) = \alpha(x)$

$\alpha(x) = \emptyset$, $\gamma(x) \neq \emptyset$, $\gamma'(y) = \gamma(x)$

$\alpha(x) = \emptyset$, $\gamma(x) = \emptyset$, $\pi'(y) = \pi(x)$

Exemple :

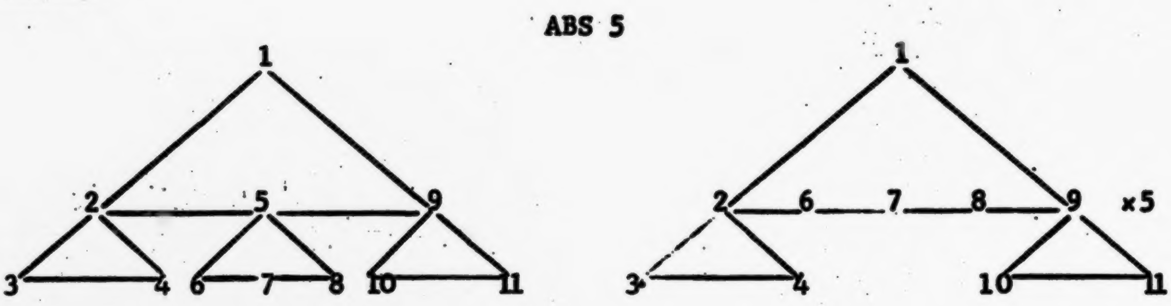


figure 13

e) Transformations .

Nous appellerons transformation une composition d'opérations de transformations élémentaires sur une arborescence donnée. Le résultat de la transformation sera une famille d'arborescences.
L'arborescence principale sera celle qui aura pour racine le même sommet que l'arborescence de départ.
Pour conserver la notion d'arborescence principale, on considère systématiquement que celle-ci est munie d'un sommet racine ⊕ sur lequel toute transformation est interdite. Pour cela ne restreigne en rien les transformations possibles, on ne lui fait jouer aucun rôle dans la structure et on lui donne un fils unique qui est la racine véritable de

l'arborescence.

Les transformations d'insertion seront notées sans y adjoindre le détachement nécessaire, implicitement défini - On notera chaque transformation de la même manière que des relations :

Exemple : " x AINE y " signifie suppression de y et insérer y comme AINE de x .

Par abus du langage, on notera de même l'échange : " x ECH y "

Et les transformations ne portant que sur un seul sommet : " $\emptyset$ DETA y "

" $\emptyset$ ABS y "

La composition de transformations élémentaires pourra se noter par une liste (programme) à exécuter.

En considérant qu'à une arborescence donnée $\mathcal{A} = (X, \Gamma_1, \Gamma_2)$ on peut associer une famille d'arborescences (Y, Γ_1, Γ_2) en prenant l'ensemble Y infini tel que $X \subset Y$ et que $\Lambda x \in Y - X, \alpha(x) = \gamma(x) = \pi(x) = \emptyset$ il sera toujours possible d'adjoindre un nouveau sommet dans l'arborescence principale par insertion. Il suffira de pouvoir définir un identificateur pour chaque sommet (on pourra par exemple, leur associer un entier : si CARD (X) = n, on notera (n+1),(n+2)... etc les sommets de Y - X).

Exemple de transformation

$$\left.\begin{array}{l}\text{6 ECH 3}\\ \emptyset\text{ DETA 3}\end{array}\right\}\begin{array}{l}\text{noté aussi 3 SUBST 6}\\ \text{substitution de z par 6}\end{array}$$

$\emptyset$ ABS 6

4 SUBST 2

11 SUBST 3

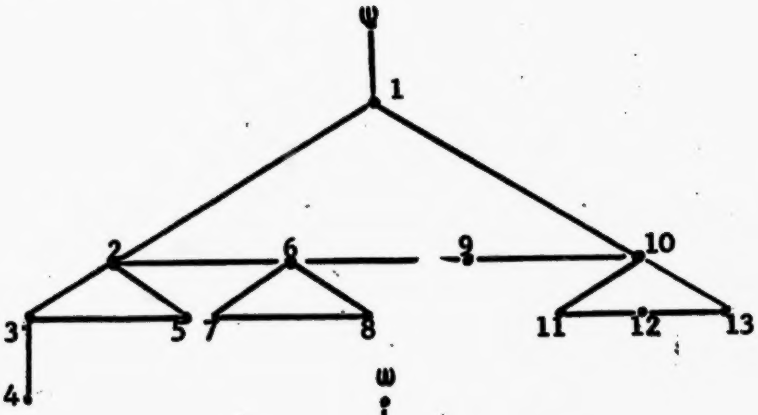
1 SUBST 11

11 AINE 6

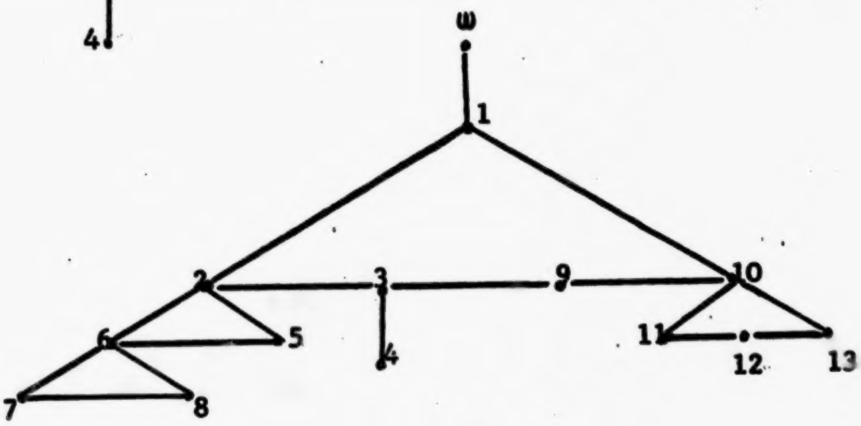
11 AINE 4

Structure de départ :

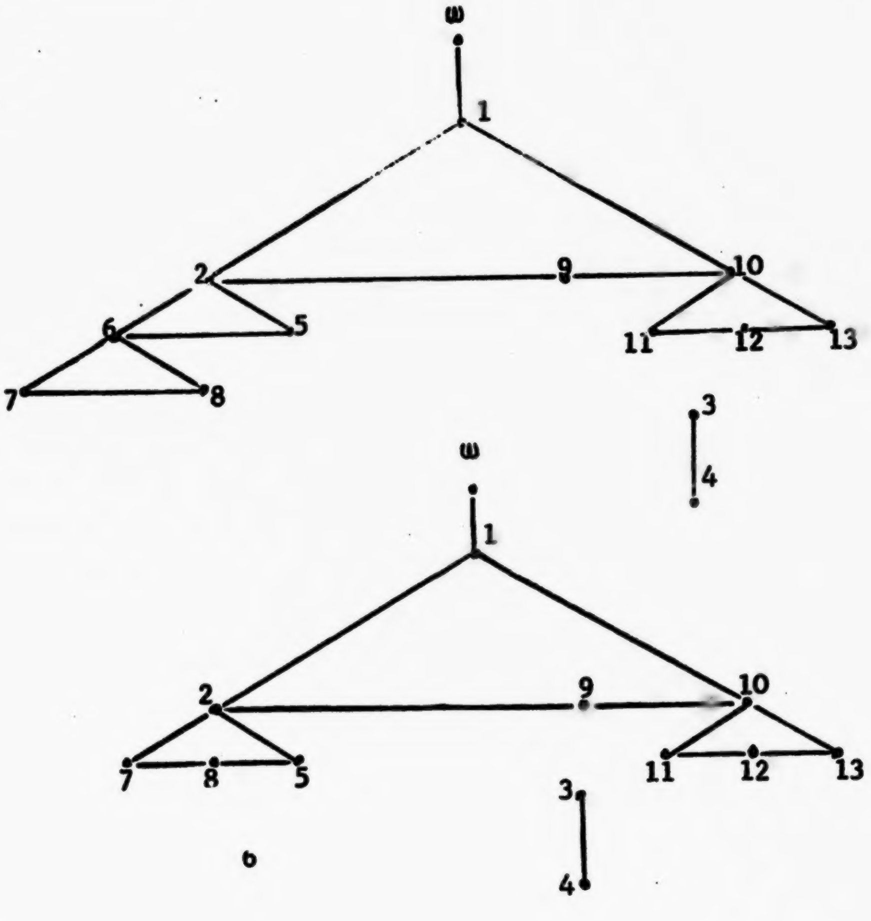
29


ω
1
2
6
9
10
3
5
7
8
11
12
13
4

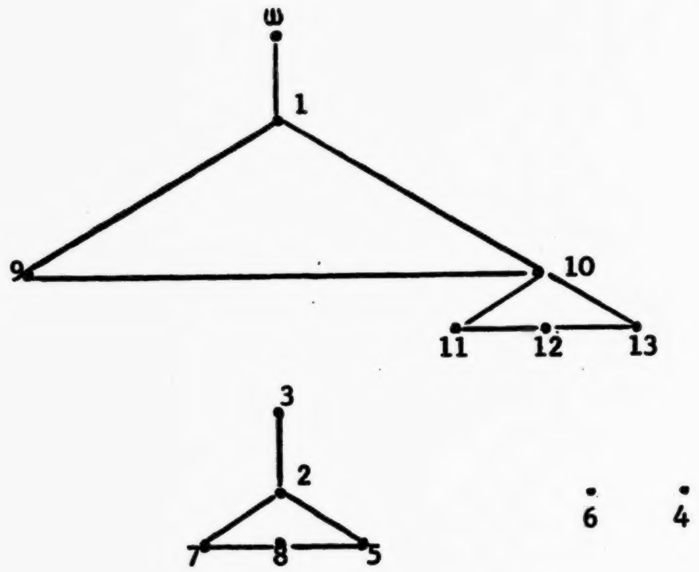
6 ECH 3


ω
1
2
3
9
10
6
5
4
11
12
13
7
8

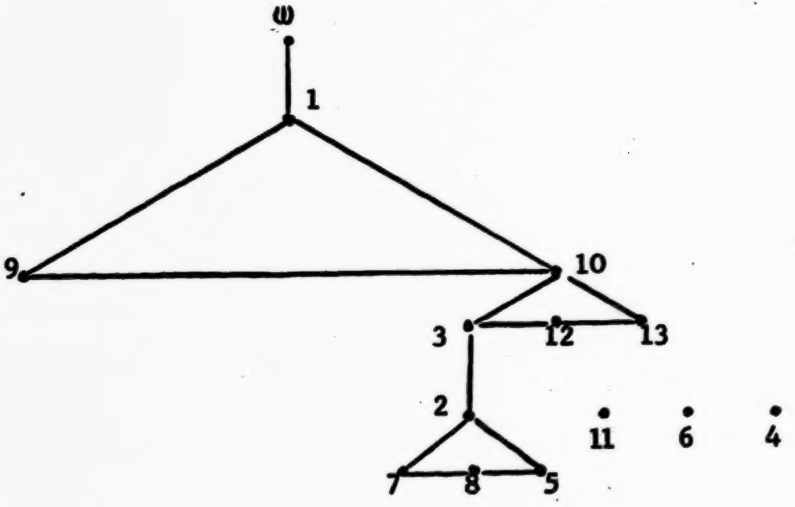
0 DETA 3


ω
1
2
9
10
6
5
11
12
13
7
8
3
4
ω
1
2
9
10
7
8
5
11
12
13
3
4
6

0 ABS 6


ω
1
9
10
11
12
13
3
2
7
8
5
6
4

4 SUBST 2


ω
1
9
10
3
12
13
2
7
8
5
11
6
4

11 SUBST 3

qui conduit à :

1 SUBST 11
11 AINE 6
11 AINE 4

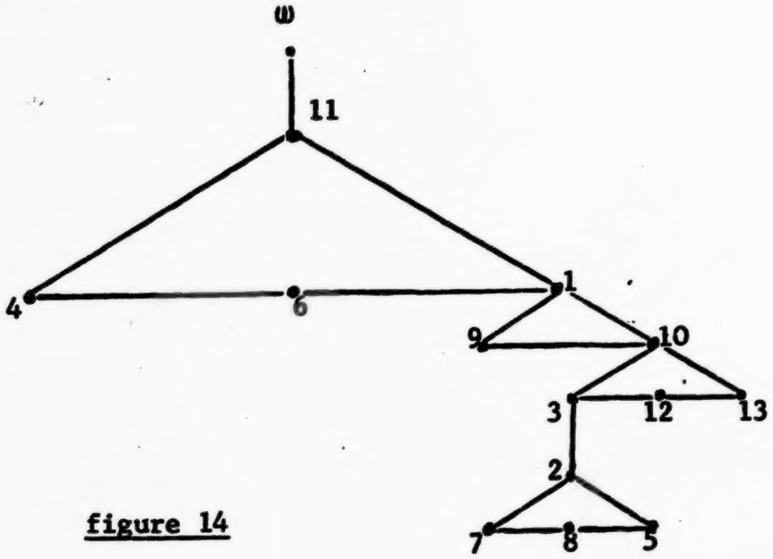

ω
11
4
6
1
9
10
3
12
13
2
7
8
5

figure 14

6° - Grammaires transformationnelles

a) Règle de transformation sur une structure :

Soit une structure $S = (\mathcal{A}, V, \Delta)$, une règle de transformation est définie par une partie gauche (identification) et une partie droite (transformation)

α) Partie gauche :

La partie gauche a pour but de repérer une figure de pivot donné dans S (4-c).
Elle est constituée par un schéma de figure F muni de symboles du vocabulaire comme il est indiqué en 4-c.
A un pivot donné, un schéma de figure peut faire correspondre plusieurs figures de la structure. Par convention, à une partie gauche de règle on associe la figure minimale conformément à l'ordre total défini en 4-b.
Soit $(x_o, x_1, \dots, x_n)$ cette figure, chacun des sommets est repéré par son indice i $(o \leqslant i \leqslant n)$.

β) Partie droite :

Elle comprend une transformation (5-e) et une application δ.
Les opérandes des transformations élémentaires sont désignés par des entiers j positifs ou nuls.
Pour tout j , $o \leqslant j \leqslant n$, les sommets sont ceux de la figure
Pour tout $j > n$, il s'agit de sommets n'appartenant pas à X (5-e)
L'application δ définit une nouvelle affectation des symboles du vocabulaire sur les sommets spécifiés en partie droite.

γ) Application d'une règle de transformation :

Soient : une structure $S = (\mathcal{A}, V, \Delta)$, de racine ω , une règle de transformation et un pivot x_o.
Le résultat de l'application de la règle est ainsi défini :

1) Si S n'admet aucune figure de pivot x_o conforme à la partie gauche, la structure est inchangée.

2) Si il existe une figure , on ne considère que celle qui est minimale, et le résultat de l'application de la règle sera une

structure $S' = (\mathcal{A}', V, \Delta')$ de racine ω . L'arborescence $\mathcal{A}'$ est obtenue par application de la transformation indiquée en partie droite.
L'application Δ' est telle que $\Delta'x = \Delta x$ si δx n'est pas définie, et $\Delta'x = \delta x$ si δx est définie en partie droite.

δ) Ecriture et exemple :

La partie gauche est notée selon les conventions définies au paragraphe 4-c.

La partie droite comporte la transformation et les affectations de symboles, notées par l'opérateur := ; il est possible de recopier le symbole affecté à un sommet sans en préciser la valeur, en notant le numéro de sommet en guise de symbole.
Reprenons l'exemple donné en 4-c

Partie gauche : * (FILS c(FILS *(t)))(FILS d COUSING c))(CADET a (!))

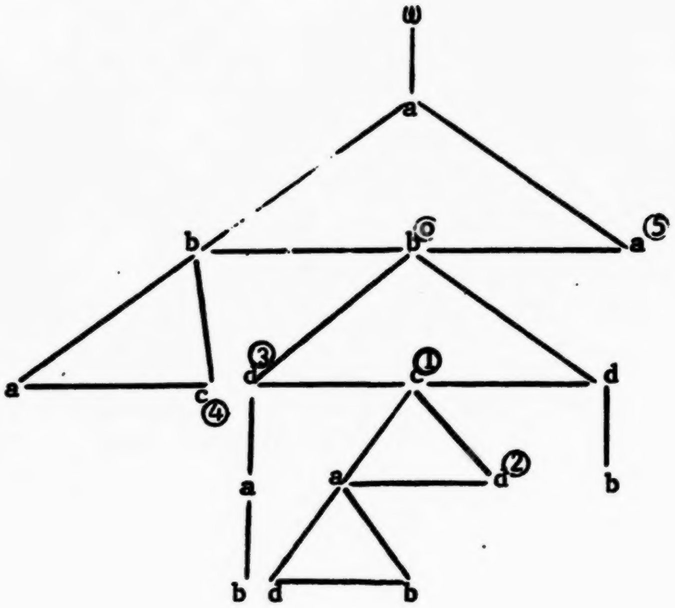


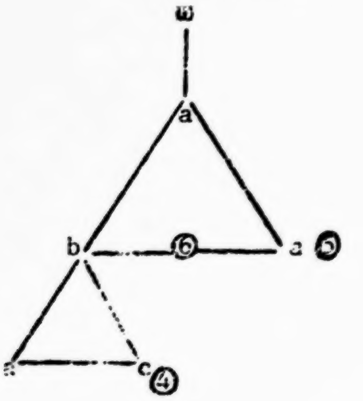
figure 15

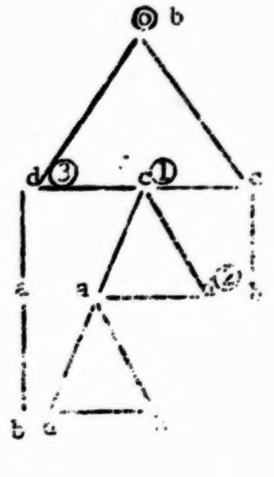
Partie droite

0 ECH 6
6 AINE 3
3 BENJ 0
ABS 0
3 CADET 1
6 := 0
3 := d
4 := a

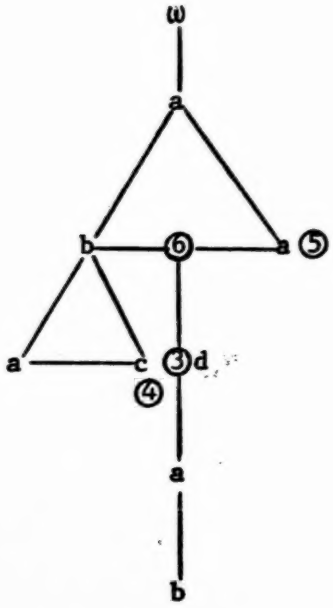
L'arborescence étant donnée, la partie gauche, avec la règle du minimum, nous sélectionne les 6 sommets marqués.
La transformation aura pour effet de recopier en FILS de 3 tous ses frères non repérés, et à lui conserver les frères repérés.

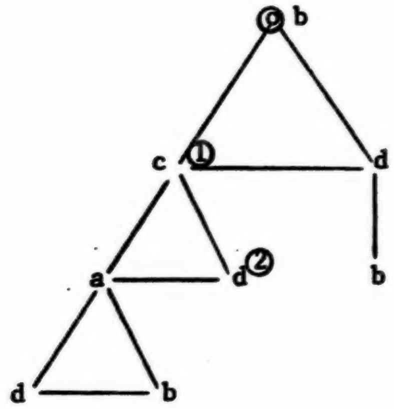
0 ECH 6 : on échange le sommet pivot et un sommet isolé : d'où deux arborescences.



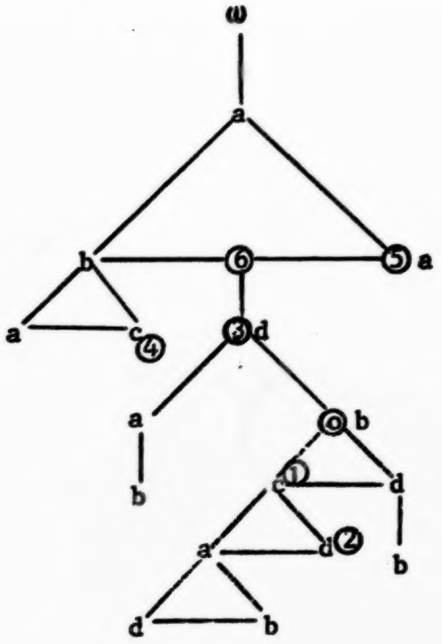


6 AINE 3 : on ramène 3 dans l'arborescence principale

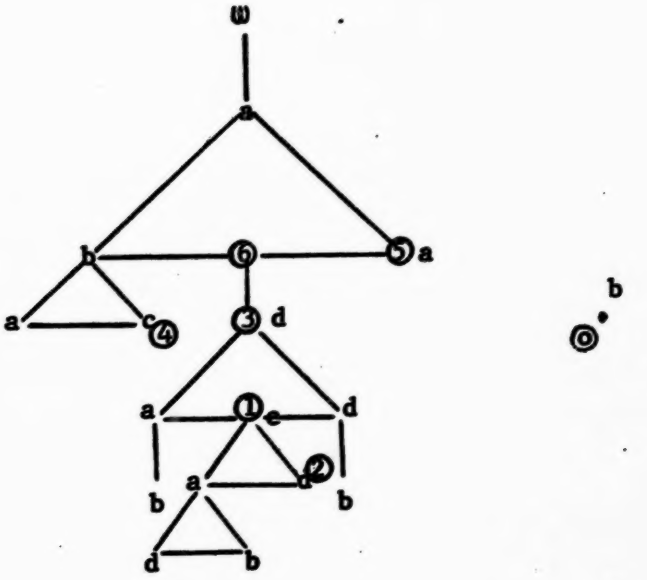

ω
a
b
⑥
a ⑤
a
c
③d
④
a
b


⓪ b
c ①
d
a
d②
b
d
b

3 BENJ 0 :


ω
a
b
⑥
⑤ a
a
c④
③d
a
b
⓪ b
c①
d
a
d②
b
d
b

ABS O :


ω
a
b
⑥
⑤ a
a
c ④
③ d
b
⓪
a
① e
d
a
d ②
b
b
d
b

3 CADET 1

6 := O

3 := d

4 := a

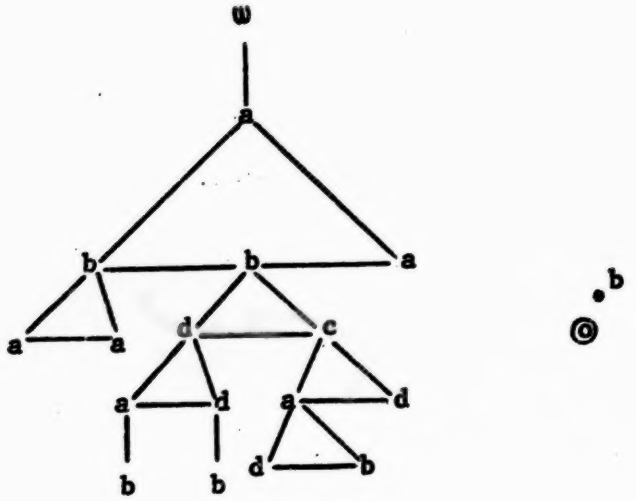

ω
a
b
b
a
b
⓪
a
a
d
c
a
d
a
d
b
b
d
b

figure 16

b) Grammaire transformationnelle :

Nous appellerons grammaire transformationnelle une liste de règles de transformation.

Une arborescence étant donnée, il peut exister plusieurs dérivations finies pour une grammaire donnée, selon le choix des pivots et l'ordre d'application des règles ; chaque dérivation finie correspond à l'obtention d'une arborescence finale stable, (sur laquelle aucune règle ne peut s'appliquer). Il est cependant possible, moyennant un certain nombre de restrictions, de définir une dérivation unique à partir d'une arborescence donnée ; nous en verrons un exemple dans l'étude des problèmes de mise en oeuvre.

II - PROGRAMMATION

Nous donnerons ici une description sommaire d'un programme permettant d'exploiter une grammaire transformationnelle.

1° - Représentation d'une structure

Un sommet est représenté par un ensemble de pointeurs (adresses) correspondant aux diverses fonctions. En pratique, compte tenu de propriétés liées au calculateur utilisé (7044 IBM), les fonctions γ et π sont représentées par un même pointeur ; en effet, $\gamma(x) \neq \emptyset \rightarrow \pi(x) = \emptyset$, $\pi(x) \neq \emptyset \rightarrow \gamma(x) = \emptyset$. A ce pointeur est associé un indicateur booléen qui définit sa nature, correspondant en fait à la propriété Benjamin. Symétriquement, il a paru commode de définir un troisième pointeur contenant γ^{-1} ou α^{-1}, et un indicateur caractérisant la propriété Ainé. Un sommet peut alors se définir par un Symbole S, un pointeur α, un indicateur B, un pointeur (γ, π), un indicateur A et un pointeur $(\gamma^{-1}, \alpha^{-1})$

avec $\bar{B}(x) \rightarrow \gamma(x) \neq \emptyset$, $\alpha(x) = \emptyset$

et $\bar{A}(x) \rightarrow \gamma^{-1}(x) \neq \emptyset$, $\alpha^{-1}(x) = \emptyset$

La racine de l'arborescence (ω) aura alors les propriétés

$$A, B, \gamma(x) = \emptyset, \gamma^{-1}(x) = \emptyset$$

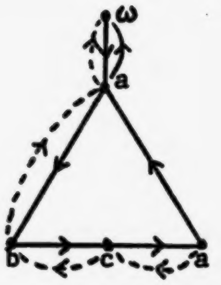
$$\pi(x) = \emptyset, \alpha^{-1}(x) = \emptyset$$

Nous représenterons schématiquement un sommet de la façon suivante :

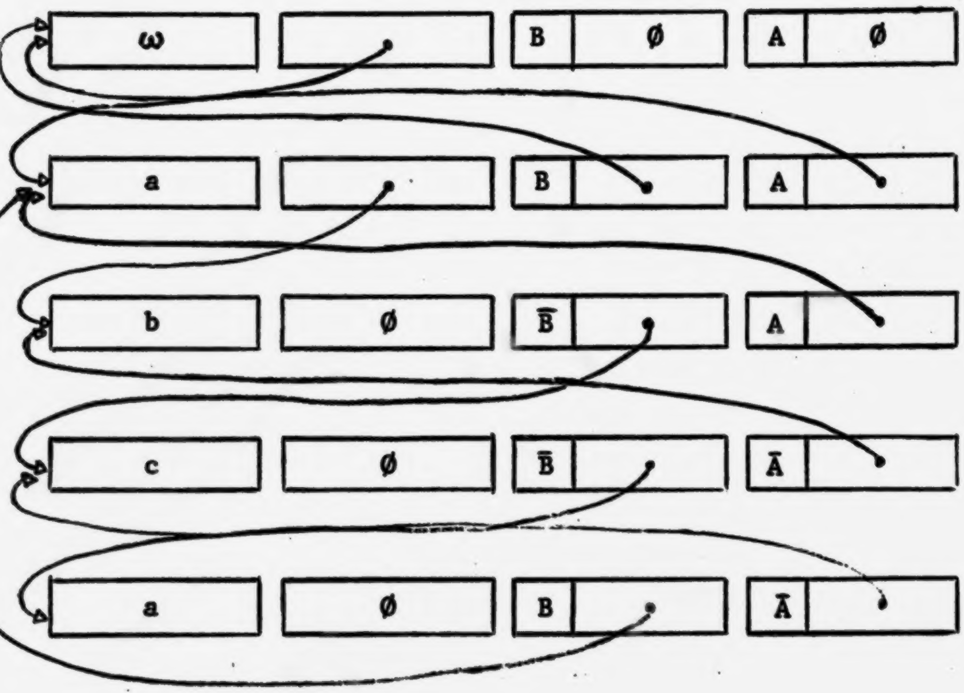
S	α	B	γ, π	A	γ^{-1}, π^{-1}

Exemple :

La structure arborescente suivante,



sera notée :



Cette représentation n'exclut pas, évidemment, l'adjonction d'autres informations sur un sommet donné, et, en particulier, de 3 pointeurs caractérisant son appartenance à une autre arborescence.

2° - Description de la reconnaissance

a) Balayage général :

La grammaire constitue en fait un programme formé par la suite séquentielle des programmes que sont les règles.

Schématiquement, le programme associé à une règle R_i recherche, pour un sommet pivot x donné, si la règle R_i est applicable.

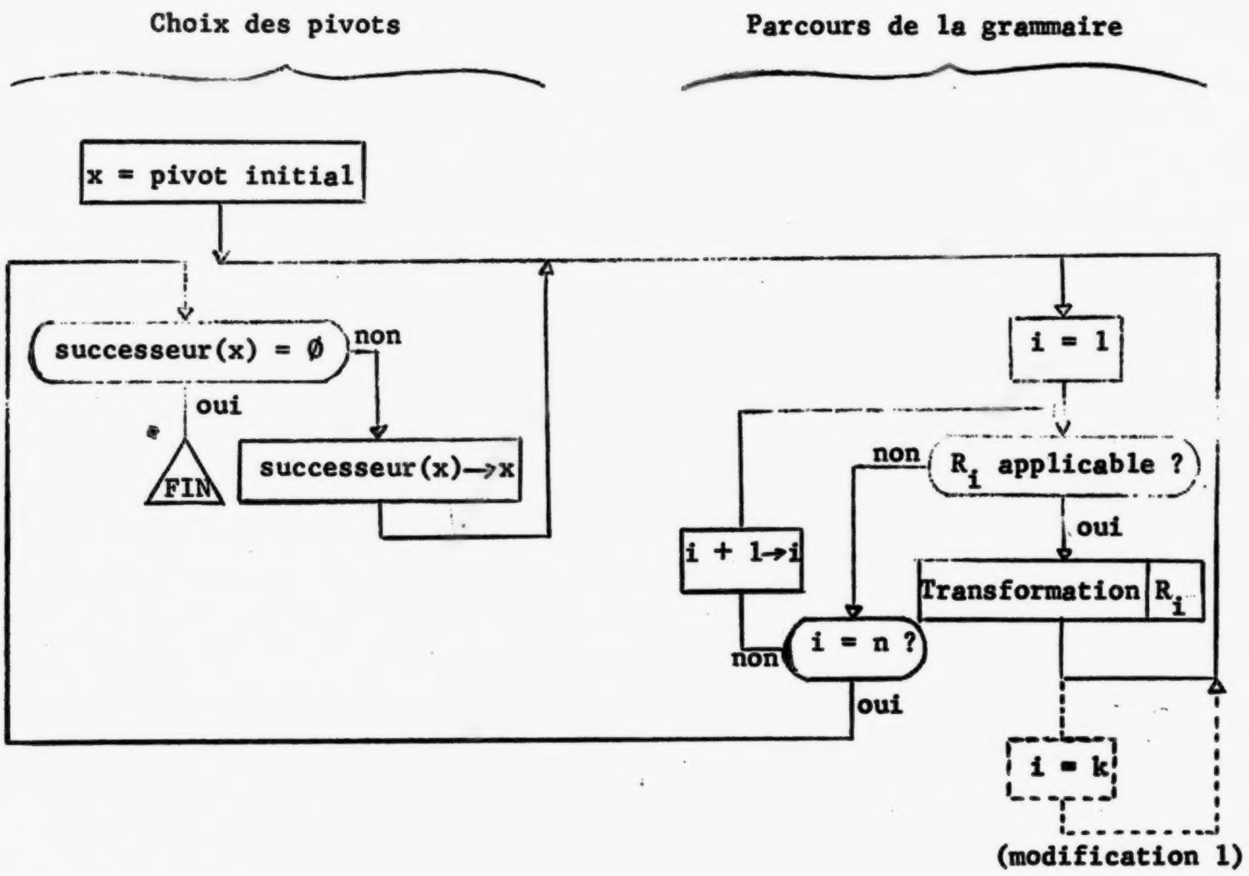
Si la règle n'est pas applicable, le même essai sera renouvelé avec R_{i+1} jusqu'à épuisement des règles de la grammaire.

Si une règle R_i est applicable, la transformation associée est réalisée et le processus complet est repris, pour le même pivot, à partir de la première règle. On voit que l'ordre séquentiel entre les règles définit un ordre de priorité d'application entre celles-ci.

Si aucune règle ne s'applique à un pivot donné, ce pivot est complètement analysé et le contrôle est donné à l'algorithme de choix des pivots, pour en déterminer un autre. Cet algorithme parcourt successivement tous les sommets de l'arborescence dans un ordre fixé (en général, l'ordre habituel de description préfixée d'une arborescence).

Plusieurs modifications peuvent accélérer et rendre plus souple le procédé, en particulier : 1) associer à chaque règle R_i le numéro de la prochaine règle à appliquer si elle est acceptée.

2) Définir des règles de changement de pivot permettant de contrôler le balayage par la grammaire.


Choix des pivots
Parcours de la grammaire
x = pivot initial
successeur(x) = Ø
non
oui
FIN
successeur(x)→x
i = 1
non
R_i applicable ?
oui
i + 1→i
Transformation
R_i
non
i = n ?
oui
i = k
(modification 1)

b) Recherche d'une figure correspondant à une partie gauche de règle donnée

Le schéma de figure est défini par une expression parenthésée contenant des noms de relations et les noms de symboles qui leurs sont associés (6-a-α).

Soit l'exemple suivant :

* (FILS a (FRERE b) (FILS c)) (FILS d)

A chaque relation correspond l'appel à un sous programme dont les paramètres sont le symbole associé et le nombre de parenthèses (n) fermées qui séparent la relation de la relation précédente.

Ainsi, l'exemple précédent se traduira par :

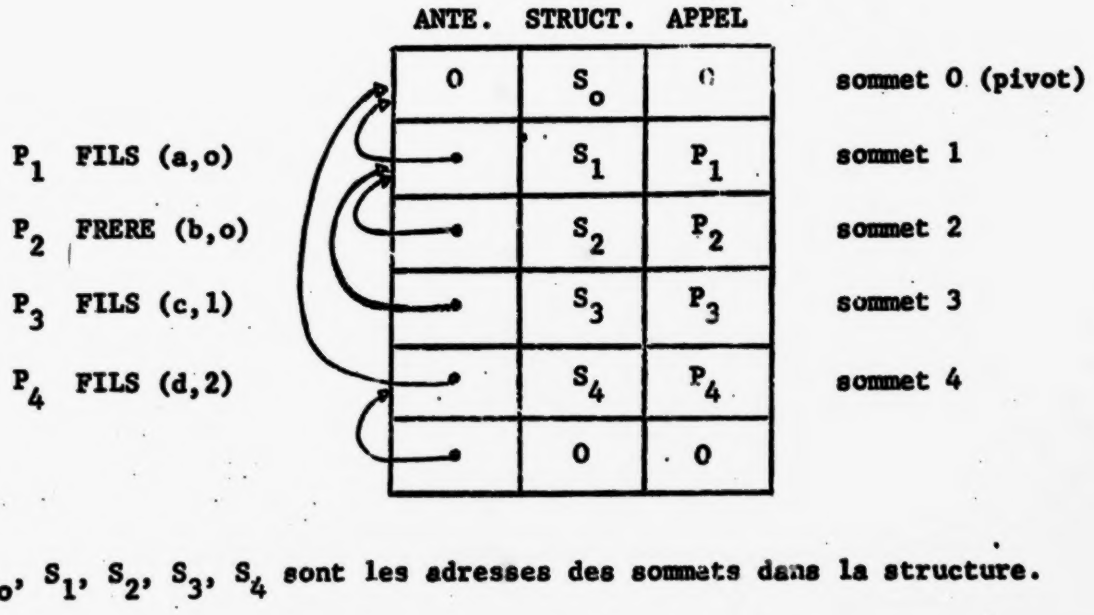
FILS (a,o)
FRERE (b,o)
FILS (c,1)
FILS (d,2)

Le sous programme relation a pour rôle de déterminer le sommet x à partir duquel on doit appliquer la relation, et de déterminer le premier y tel que xRy et que le symbole associé soit égal au symbole demandé.

En pratique, le déroulement du programme utilise une zone de mémoire, dite Pile de contrôle, dans laquelle sont stockées des informations relatives à la règle en cours d'application. A chaque sommet y sélectionné par la règle correspond une adresse dans la pile et un groupe de trois pointeurs : un pointeur vers le sommet x à partir duquel y a été obtenu (Antécédent), un pointeur vers l'adresse exacte de y dans la structure, et un pointeur contenant l'adresse d'appel du sous programme qui a localisé y.

Si P_1, P_2, P_3, P_4 sont ces adresses dans l'exemple donné, nous aurons :

Pile de contrôle



	ANTE.	STRUCT.	APPEL	
	0	S_o	0	sommet 0 (pivot)
P_1 FILS (a,o)	•	S_1	P_1	sommet 1
P_2 FRERE (b,o)	•	S_2	P_2	sommet 2
P_3 FILS (c,1)	•	S_3	P_3	sommet 3
P_4 FILS (d,2)	•	S_4	P_4	sommet 4
	•	0	0	

où S_o, S_1, S_2, S_3, S_4 sont les adresses des sommets dans la structure.

La référence ANTE est mise à jour à chaque nouvelle relation en utilisant le nombre n de parenthèses fermées. Elle est systématiquement initialisée, en admettant n = 0, à la valeur correspondant au sommet précédent dans l'ordre séquentiel.

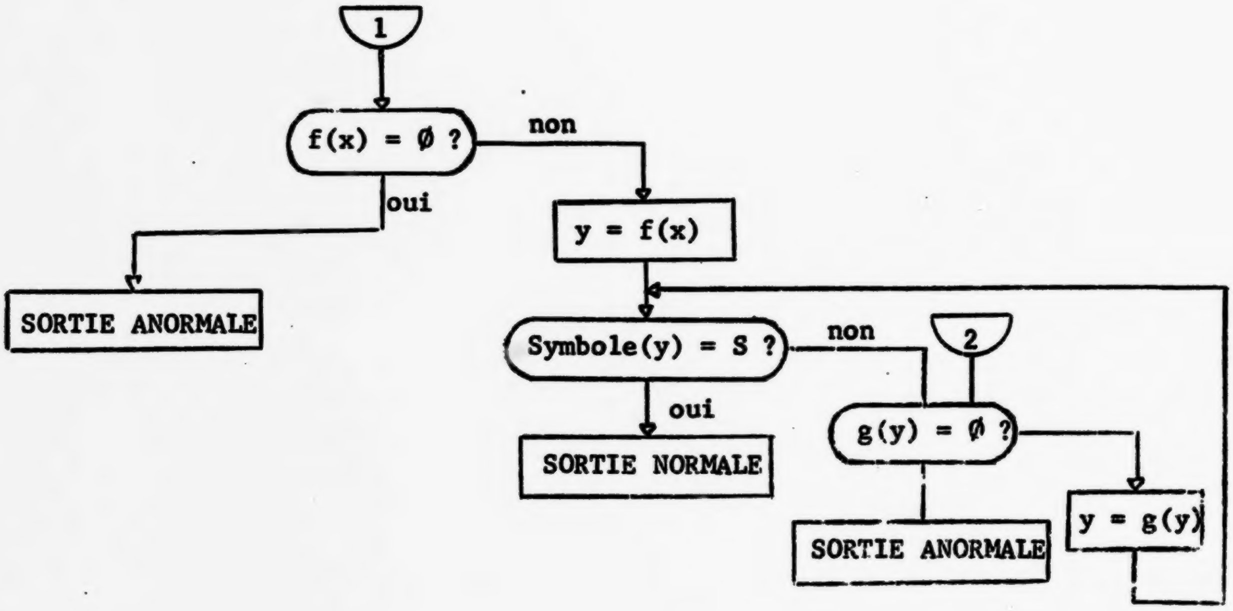
Ainsi, un éventuel sommet S_5 aurait déjà S_4 pour antécédent si n = 0, et si n = 1.

La référence APPEL est nécessaire pour la recherche systèmatique. Ainsi, si l'on avait obtenu S_1 et S_2 à partir de S_o, et si l'on ne trouvait pas de sommet S_3, il faudrait reprendre la recherche à partir d'un autre sommet S'_1.

De S_3, ANTE permet de retrouver S_1, et de réappliquer la relation P_1 au-delà de S_1.

On sait que chaque relation est définie par deux fonctions f et g.

Un organigramme approximatif serait le suivant :



L'entrée 1 correspond au premier appel de la relation. L'entrée 2 correspond au cas où l'on revient sur la relation donnée pour déterminer un autre chemin.

L'entrée 2 fait suite à une sortie anormale dans une autre relation dont le sommet a pour antécédent le sommet dont on repart en 2. Dans l'exemple, P_1 sera appelée par l'entrée 1 au départ, ainsi que P_2 et P_3. Si P_3 donne une sortie anormale, on reprendra P_1 par l'entrée 2, puis à nouveau P_2 et P_3 par l'entrée 1.

3° - Application des transformations

Cette partie ne présente pas de difficultés majeures. A chaque transformation élémentaire correspond un sous programme, dépendant de 3 paramètres en général :

- Le numéro d'ordre du sommet y concerné par la transformation
- Le numéro d'ordre du sommet permettant d'affecter une position à y (ce numéro est vide pour un détachement ou une absorption).
- Le nouveau symbole S' à affecter à y.

Les numéros d'ordre sont utilisés comme des références dans la pile utilisée en partie gauche. La définition d'un nouveau sommet se fait en lui attribuant un numéro d'ordre supérieur à ceux prévus dans le schéma de figure. On obtient ainsi une référence dans la pile à laquelle il ne correspond aucun sommet, ce que l'on interprête en adjoignant un nouveau sommet à la structure.

III - APPLICATION AUX PROBLEMES DE TRADUCTION AUTOMATIQUE

1° - Le rôle des grammaires transformationnelles dans le système de T.A.

L'objectif du modèle de reconnaissance syntaxique (M2), ainsi que la grammaire et les algorithmes qui le réalisent ont fait l'objet d'une publication antérieure [6] [8]

Rappelons cependant que à chaque phrase du texte initial, le modèle M2 fournit une ou plusieurs structures qui se caractérisent de la manière suivante : c'est une structure arborescente dans laquelle à chaque occurrence correspond un sommet ; chaque symbole du vocabulaire V comprend le nom d'une règle de construction syntaxique r (telle que $BC \succ\!\!- A$) et un ensemble d'informations caractérisant l'unité lexicale (variables grammaticales, codes de référence au lexique, etc...)

Entre deux sommets x et y la relation x FILS y a lieu si et seulement si les occurrences associées sont liées par une règle syntaxique r dont x est le gouverneur. Alors r fait partie du symbole de y. En outre, même si l'algorithme de réduction des constituants discontinus a dû être employé dans la phase M2, l'ordre entre les fils d'un même père reste défini par l'ordre sur les occurrences.

Exemple :

Nous traiterons complètement la phrase suivante :

#ЭТА ФОРМУЛА ОБЬЯСНЯЕТ ЦАСТОЕ ВОЗНИКНОВЕНИЕ ДЕЙТРОНОВ

ЯДЕРНЫХ РЕАКЦИЯХ.

(# cette formule explique fréquente apparition [de] deuterons dans [les] nucléaires réactions).

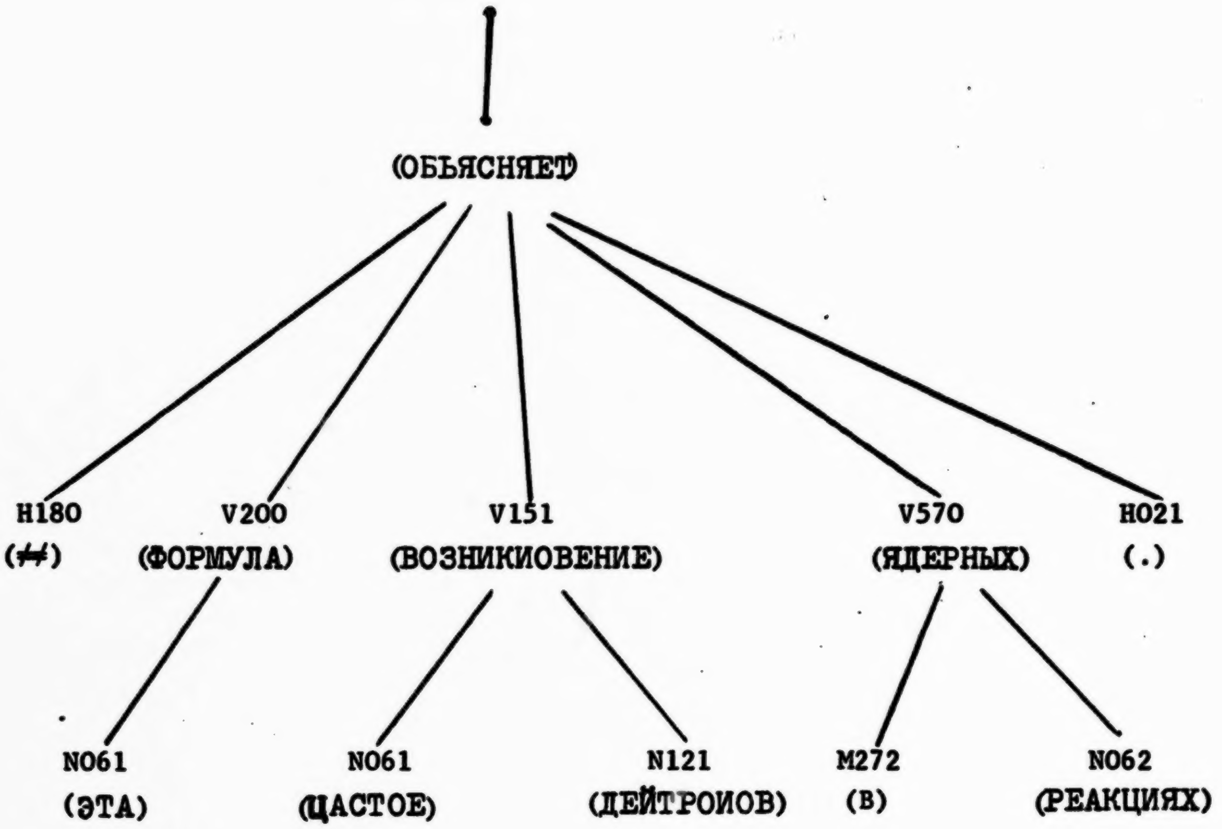

(ОБЬЯСНЯЕТ)
H180
(≠)
V200
(ФОРМУЛА)
V151
(ВОЗНИКИОВЕНИЕ)
V570
(ЯДЕРНЫХ)
H021
(.)
N061
(ЭТА)
N061
(ЦАСТОЕ)
N121
(ДЕЙТРОИОВ)
M272
(В)
N062
(РЕАКЦИЯХ)

figure 17

En réalité, la structure indiquée représente deux structures compatibles avec la grammaire de M2 :

- elle-même
- celle dans laquelle le groupe prépositionnel est gouverné par ВОЗНИКНОВЕНИЕ (apparition), qui est la structure correcte.

A chaque structure ainsi définie on superpose une autre structure arborescente : la structure de chaîne des occurrences.
Il sera alors possible d'écrire des schémas de figure comportant des relations sur l'une quelconque de ces deux structures ; De plus, le chemin entre deux sommets exprimé par une relation peut être une composition des fonctions α, γ et π sur des structures différentes.

Le passage à la construction syntaxique française se fait au moyen d'une structure intermédiaire (dans le langage pivot) ce qui implique l'emploi de deux modèles transformationnels successifs.

2° - Application au modèle M3

L'objectif du modèle d'étiquetage (M3) consiste à :

- Déterminer la structure syntaxique correcte à partir de la famille de structures implicitement proposées au moyen de transformations sur la structure arborescente initiale.

- Substituer aux noms de règles syntaxiques une étiquette caractérisant la signification de la liaison entre deux mots de manière à éliminer les propriétés formelles de la langue source.

- Supprimer les sommets (occurrences) qui sont relatives à des mots considérés comme outils syntaxiques dans la langue source (par exemple : certaines ponctuations, prépositions fortement régies, etc...)

- Dupliquer certains sommets lorsque les occurrences associées possèdent plus d'une fonction syntaxique.

- Créer de nouveaux sommets lorsque la langue emploie une tournure synthétique.

La phrase de l'exemple précédent va être représentée dans M3 par la structure donnée en figure.

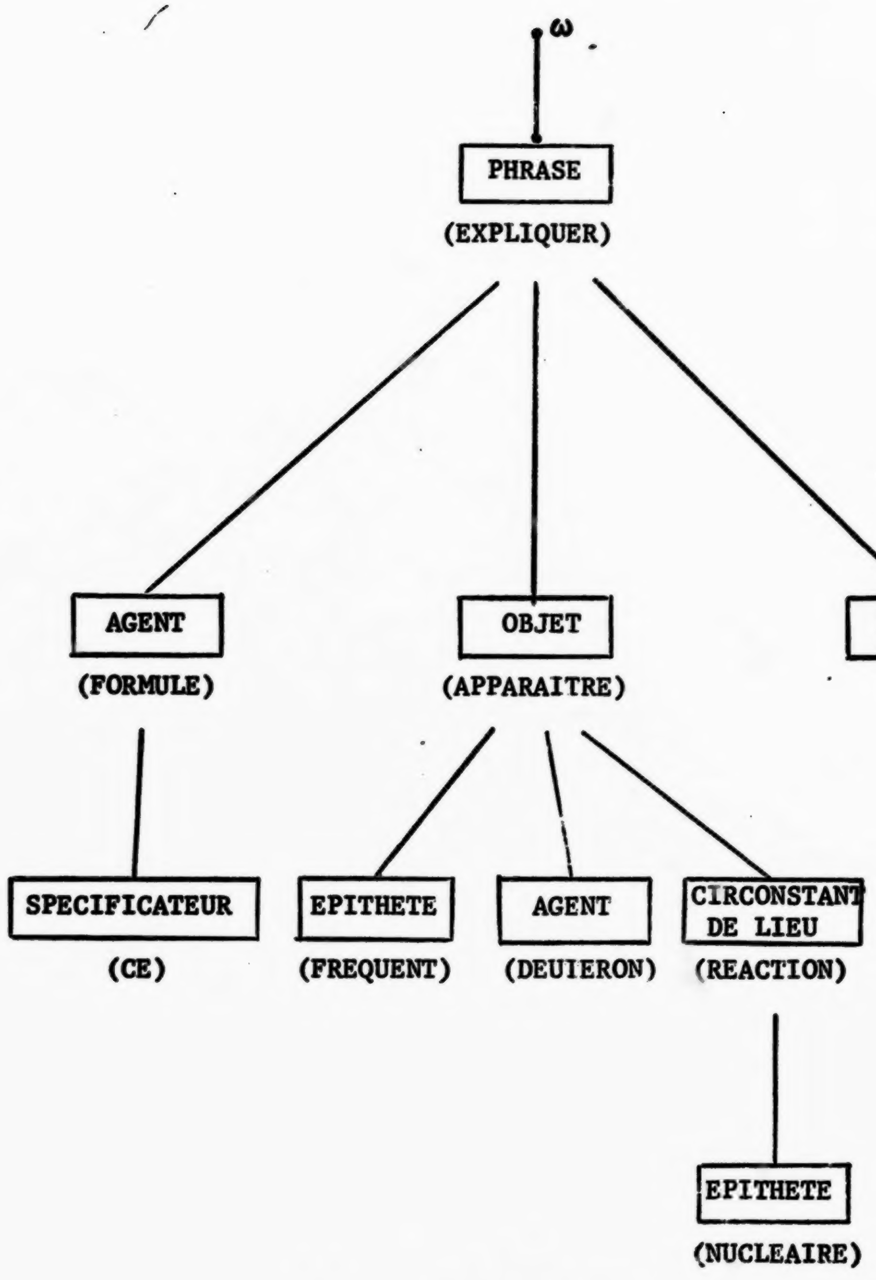

ω
PHRASE
(EXPLIQUER)
AGENT
(FORMULE)
OBJET
(APPARAITRE)
FIN DE PHRASE
(.)
SPECIFICATEUR
(CE)
EPITHETE
(FREQUENT)
AGENT
(DEUTERON)
CIRCONSTANT DE LIEU
(REACTION)
EPITHETE
(NUCLEAIRE)

figure 18

Comme chaque symbole du vocabulaire porté par la structure initiale comprend un nom de règle syntaxique et une quantité appréciable d'informations, il est préférable de ne considérer d'abord que le nom de règle pour la recherche d'une figure obéissant à un schéma donné. Les informations composant le reste du symbole seront traitées en tant que valeurs de variables associées à la classe de symboles représentés par le même nom de règle.

Ainsi la partie gauche d'une règle de transformation se divise en deux sections :

- Un schéma de figure
- Des valeurs de variables sur les sommets de la figure identifiée, et, éventuellement, de nouvelles relations faisant apparaître de nouveaux sommets.

Exemple : * (FILS N121) est un schéma de figure sur la structure arborescente qui constitue la première section

a : NON-REFLECHI ∩ INTRANSITIF
b : REFLECHI ∩ INTRANSITIF
c : TRANSITIF-CONSTANT
d : NON-REFLECHI ∩ TRANSITIF-VARIABLE
e : REFLECHI ∩ TRANSITIF-VARIABLE
f : QUALITE
g : UNITE
h : ANIME ∪ PERSONNIFIE
i : INANIME ∩ NON-DENOMBRABLE

} qui constituent la deuxième section

Selon différents assemblages de conditions sur les sommets de la figure on peut imaginer que des transformations différentes doivent être effectuées.

Aussi, la règle de transformation peut elle comporter plusieurs parties droites "conditionnelles".

La continuaticn du même exemple fournit les parties droites suivantes

- (a [o]) (d [o] ∩ h [1] ⟹ 1 := AGENT
- (b [o]) ∪ (e [o] ∩ h [1] ⟹ 1 := AGENT
  2 := OBJET (PRONOMINAL-REFLECHI)
  O AINE 2
- (c [o]) ∪ (d [o] ∩ i [1]) ∪ (d [o]) ∪ (e [o]) ⟹ 1 := OBJET
- f [o] ⟹ 1 := SUBJECT
- (g [o] ∩ i [1] ⟹ 1 := MATIERE
- * ⟹ 1 := DETERMINATIF

Dans une telle règle la figure comporte 2 points : le premier point noté 0, le deuxième point noté 1 doit être fils du point 0 et porter le numéro de règle N121.

On trouve une telle figure dans l'exemple traité en partant du point correspondant au mot ВОЗНИКНОВЕНИЕ

Le point o est donc ВОЗНИКНОВЕНИЕ

Le point 1 est alors ДЕЙТРОНОВ

On obtient alors l'étiquette "AGENT" pour ДЕЙТРОНОВ qui est en effet l'agent de l'action "apparaître".

Dans le cas où le deuxième groupe de propriétés est réalisé, on construit un point supplémentaire noté 2 auquel on affecte l'étiquette "OBJET (PRONOMINAL-REFLECHI)" et on le place dans la structure en tant que "fils ainé" du point 0.

L'algorithme d'exploitation de la grammaire cherche à traiter les figures issues de la racine de la structure initiale, puis progresse par "noyaux" successifs. Les règles de la grammaire sont ordonnées. Une dérogation à l'ordre de progression dans les noyaux successifs a lieu dans deux cas :

- On rencontre un mot faisant partie d'une tournure idiomatique (priorité absolue)
- On rencontre une coordination (priorité relative)

Lorsqu'il ne reste plus des points "étiquetés" l'algorithme de transformation s'arrête.

BIBLIOGRAPHIE

[1] - N. CHOMSKY - Aspects of the theory of syntax (1965)
The M.I.T. Press

[2] - Z. HARRIS - String analysis of sentence structure (1962)
MOUTON & C°

[3] - E. S. KLIMA - Relatedness between grammatical systems
Language (1964) volume 40 n° 1

[4] - S. KUNO - A system for transformational analysis
(Congrès NEW-YORK - mai 1965)

[5] - PETRICK - A recognition procedure for transformational grammars

[6] - B. VAUQUOIS
G. VEILLON
J. VEYRUNES - Syntaxe et Interpretation
(Congrès NEW-YORK - mai 1965)

[7] - HOLT - A mathematical and applied investigation of tree structures for computer syntactic analysis
Thèse Univ. of Pennsylvania

[8] - G. VEILLON
J. VEYRUNES - Etude de la réalisation pratique d'une grammaire "Context-Free" et de l'algorithme associé
Document G-2001-1 - avril 1964

21

A computer model for Russian grammatical description, and a method of English synthesis in machine translation

D.M. Yates (National Physical Laboratory, Teddington)

Introduction

This paper is the second of two from the NPL MT group at this conference. It describes a model designed to express the grammatical facts discovered by the Russian analysis algorithm in such a way that they can be used directly by the English synthesis algorithm. The general nature of this synthesis process is the subject of the second part of the paper.

The model: linguistic features

Russian and English have many important categories in common. For instance, both have subjects, verbs, objects, nominal groups, conditional clauses and so on. When it comes to finer details, though, the differences between the two languages become more noticeable than the similarities: the use of auxiliary verbs to represent tenses, for instance, is quite different (e.g. did not ask = не спросил)

The basic task of this model is to provide a means of representing in the computer any Russian grammatical structure which the analysis algorithm may have to express. As far as possible this representation must be independent of the particular conventions of either language. For example не спросил would not be ascribed any internal structure, but would be represented as "спрос-/ask, negative, past tense". The analysis would discover these facts, concerning itself only with Russian conventions, and the synthesis would express them in English, concerning itself only with the English conventions. "Negative" and "past tense" are examples of choices within closed sets of possibilities. Such sets are known as systems. Our model therefore has two main linguistic features, structure and system, which will both be needed to describe a Russian sentence. This terminology is taken from the work of Halliday (1961).

The structure is fundamentally a hierarchy of constituents, but there are four ways in which it differs from a conventional constituent structure:

(1) Each constituent may exemplify choices in systems, and, as illustrated above, this means that some units in the text (e.g. particles and auxiliaries) are not given places in the structure.

(2) One item may occupy more than one place in the structure. The only need for this in scientific Russian seems to be

the dual role of a relative word in linking a subordinate clause to some higher constituent and at the same time taking some role within its structure.

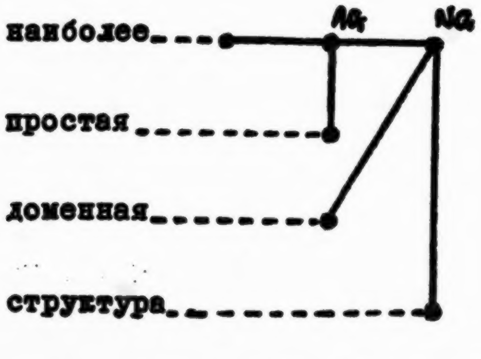
(3) There is no requirement for a constituent to be continuous in the text (although those found by the current analysis algorithm always are).

(4) If the systems are powerful enough there is no need for explicit ordering of subconstituents. This point will be taken up again later.

The model: computing features

A grammatical structure of word-groups is represented in the computer by a list structure, that is to say a collection of stored items called elements with the property that each element either (i) contains addresses of one or more other elements, or (ii) is marked as a terminal element. The terminal elements represent single items (words or idioms); the other elements each represent a larger word-group or constituent of the sentence. If an element A contains the address of an element B, this represents the fact that word-group A includes word-group B.

For example, the description of the structure of the group наиболее простая доменная структура includes four terminal elements (for the four words) and two other elements, linked as follows:

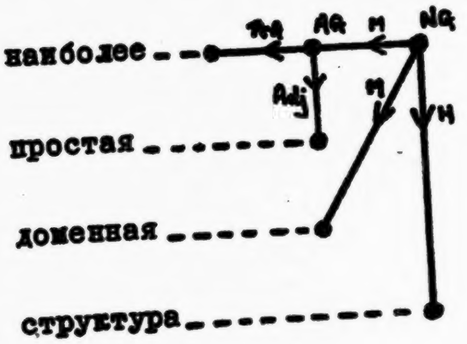


(AG = adjectival group,

NG = nominal group)

Each element is labelled with a code giving the constituent type (noun, verbal group, etc.), and each address referring to an included word-group is labelled with a code giving the role of the smaller group in the larger one (complement in prepositional group, for instance).

With roles included, the above description becomes:



Roles:
M = modifier (in NG)
H = head (in NG)
PrA = pre-adjective (in AG)
Adj = adjective (in AG)

Choices in systems are also represented in a label in the element concerned. This label is called the systems word. In the above example, the systems word in the nominal group element records the number, gender, and case of the group.

In theory, the observed order of items is either evidence for a particular structure (as in the order of prepositions and their complements), or evidence for a choice in a system (as in the order of auxiliary and subject in English interrogative sentences). Just the same is true of punctuation (some commas indicate structure, e.g. those marking clause boundaries; others indicate a choice in a system, e.g. those distinguishing 'descriptive' and 'restrictive' qualifiers in nominal groups). Ideally then the model would have no need to represent item order or punctuation explicitly: it would record the structures and systems, and the synthesis algorithm would have a free hand in determining the English order and punctuation according to English structural and systemic rules. But in practice the language features concerned are not yet understood in sufficient detail, so the synthesis keeps the original order and punctuation except where it has some reason to change them. This means that they need to be recorded in the model statement. The addresses in an element are therefore stored in the same order as the constituents to which they refer, and each element includes details of any punctuation surrounding the constituent.

The full list of constituent types and roles is as follows:

Constituent	Subconstituents' roles
Nominal group (NG)	Modifier (M) Head (H) Qualifier (Q) Appositive (Ap)
Adjectival group (AG)	Pre-adjective (PrA) Adjective (Adj) Post-adjective (PtA)
Prepositional group (PG)	Preposition (Pp) Complement (Ct)
Adverbial group (ADV)	Pre-adverb (PrA) Adverb (Adv) Post-adverb (PtA)
Verbal group (VG)	Verb (V) Complement (Ct) Adjunct (At)
Coordinate group (CG)	Conjunction (Cj) Member (Mb)
Clause (CL)	Subject (S) Predicate (Pd) Adjunct (At)
Subordinate clause (SC)	Conjunction (Cj) Clause (Cl)
Complex clause (CC)	Clause (Cl) Adjunct (At)
Comparative group (CPG) (e.g. как + noun)	Link (Lk) Comparison (Cp)
Prefix group (PFG) (e.g. вектор-функций)	Prefix (Pf) Stock (St)

Although most of the terminology in the table will be self-explanatory, it should be made clear that in a co-ordinate group the 'members' may be constituents of any type. Likewise the prefix group is a general one, the 'stock' being noun, adjective, or verb. (In practice, for reasons of programming convenience, the prefix group was not used, such groups being represented by the 'stock' alone, tagged with the reference number of the prefix).

The full list of constituent types and roles is as follows:

Constituent	Subconstituents' roles
Nominal group (NG)	Modifier (M) Head (H) Qualifier (Q) Appositive (Ap)
Adjectival group (AG)	Pre-adjective (PrA) Adjective (Adj) Post-adjective (PtA)
Prepositional group (PG)	Preposition (Pp) Complement (Ct)
Adverbial group (ADV)	Pre-adverb (PrA) Adverb (Adv) Post-adverb (PtA)
Verbal group (VG)	Verb (V) Complement (Ct) Adjunct (At)
Coordinate group (CG)	Conjunction (Cj) Member (Mb)
Clause (CL)	Subject (S) Predicate (Pd) Adjunct (At)
Subordinate clause (SC)	Conjunction (Cj) Clause (Cl)
Complex clause (CC)	Clause (Cl) Adjunct (At)
Comparative group (CPG) (e.g. как + noun)	Link (Lk) Comparison (Cp)
Prefix group (PFG) (e.g. вектор-функций)	Prefix (Pf) Stock (St)

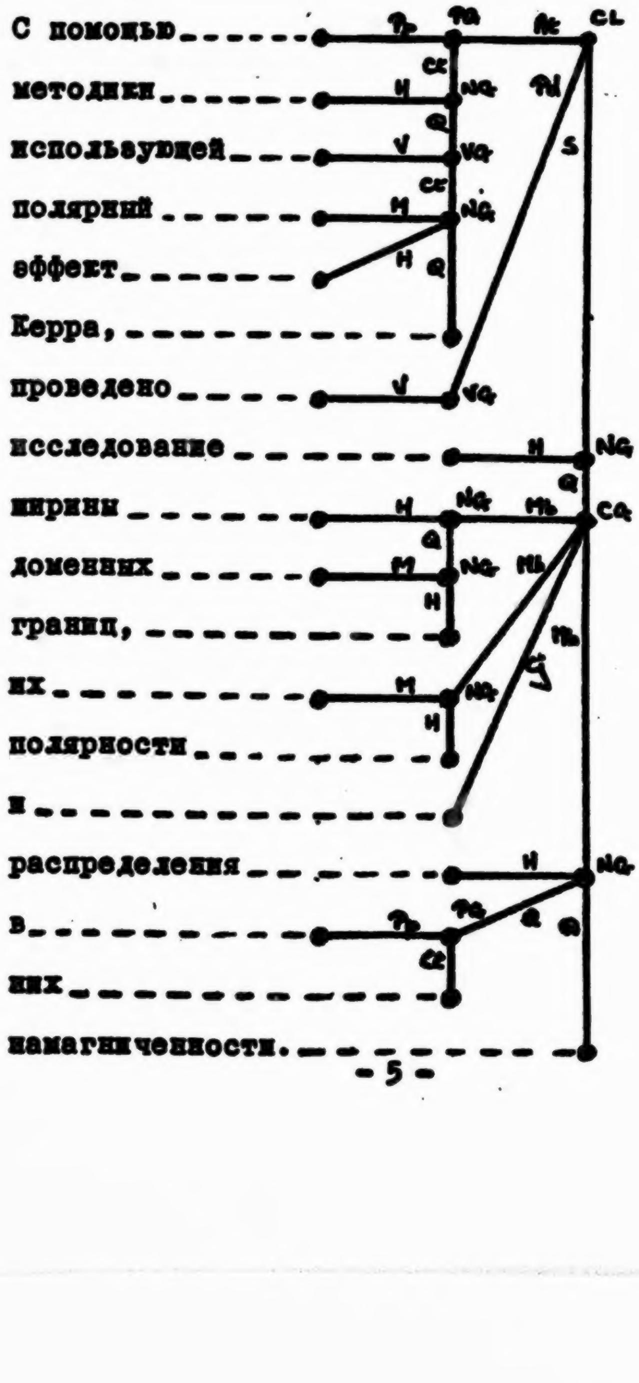
Although most of the terminology in the table will be self-explanatory, it should be made clear that in a co-ordinate group the 'members' may be constituents of any type. Likewise the prefix group is a general one, the 'stock' being noun, adjective, or verb. (In practice, for reasons of programming convenience, the prefix group was not used, such groups being represented by the 'stock' alone, tagged with the reference number of the prefix).

The table attempts to provide an adequate set of constituent types and roles for the description of sentences in our texts. It should not be inferred that our analysis processes could recognise all these features; indeed the clauses and the comparative group were not used at all.

Associated with each type of constituent there are certain systems. For example, a clause may be either non-finite (если импульс подать ...) or finite. If finite, choices of mood (interrogative/imperative/declarative), conditionality, and personality will have been made; and if the clause is personal there will be selections of person and number. All these systemic choices would be recorded in the element representing the clause.

Below, an example is given of the structural description of a complete sentence; again it is not a structure which the current analysis could produce, but is intended simply to illustrate the use of the model.

Example of sentence structure description


С помощью
методики
использующей
полярний
эффект
Керра,
проведено
исследование
ширины
доменных
границ,
их
полярности
и
распределения
в
них
намагниченности.
Pp
PG
Ac
CL
Cx
H
NG
Pd
Q
V
VG
S
M
CG
Mb

(N.B. С помощью is treated as one item since it is included in the dictionary as an idiom).

The English synthesis algorithm

The synthesis algorithm has the task of taking a sentence expressed in terms of the model described above, and producing from it the string of characters which form the English output sentence.

The program uses the model statement to guide it in decisions on:

(1) re-ordering;

(2) insertion of English 'function' words (auxiliary verbs, etc.);

(3) selection of English equivalents from the short list in each dictionary entry;

(4) inflection of English equivalents.

These decisions are of course based on grammatical data only (both structural and systemic); in particular in the selection of equivalents no semantic or collocational techniques are used.

The particular tasks under these headings which are appropriate to a particular type of constituent will in general need to be carried out whatever the role of the constituent in some higher structure may be; and we are therefore led to the need for a separate routine for each constituent type. Such a routine will be called a constituent type procedure (CTP). The nominal group CTP, for example, will be called upon when and only when a nominal group has to be produced by the program.

Since constituents nest within one another freely, one CTP will need to call on others to deal with the parts of the constituent in turn. The CTPs must in fact be written as fully recursive subroutines; and the program consists basically of a control routine for exploring the list structure together with a set of CTPs, one for each constituent type.

As was pointed out by Yngve (1960), it is a linguistic fact (at least in the Indo-European family of languages to which Russian and English both belong) that in many constituents the final sub-constituent is a group of words, while other sub-constituents are more frequently single items. Thus multiple "nesting" of the CTPs usually involves final subconstituents. But in these cases all details of the higher constituent can be "forgotten" by the computer since that constituent will not need to be returned to; so even a long sentence needs no great depth of push-down store to handle the nested CTPs. (Language has presumably evolved in this way because of an analogous advantage

---

Trois modes de consultation paraissent adéquats selon le rythme et le volume des questions.
" En direct " tout d'abord, le programme se trouve en permanence tout entier dans la machine, munie de tous les fichiers, et simultanément d'ailleurs avec des programmes appelés pour d'autres travaux. Le demandeur possède sur sa table un clavier d'appel qu'il actionne à tout moment et même pour une seule question. La réponse apparaît sur un écran de télévision ou se dactylographie sur une machine à écrire. Nous arriverons à ce mode de fonctionnement lorsque notre ordinateur sera organisé en multiprogrammation et lorsque nous posséderons des unités de stockage permanent pour nos dictionnaires.
L'organisation actuelle permet de travailler " en différé ", c'est-à-dire de réunir jusqu'à mille phrases demandées en une journée ; les faire perforer par une dactylo sur des cartes IBM et de distribuer les réponses quelques heures plus tard.

Mais une organisation bien plus efficace, lorsqu'il s'agit de traduire systématiquement de grands volumes de textes, pourra être mise en oeuvre bientôt. Il existe déjà en Europe des imprimeries qui font corriger et justifier automatiquement en ordinateur les textes à imprimer. Au moment où l'imprimeur passe les textes de la bande linotype en ordinateur nous pouvons les introduire et les conserver sur bandes magnétiques sans avoir à les perforer. Dès lors si ces textes sont destinés à être traduits, nous pouvons les passer sans aucun travail humain supplémentaire sur notre programme de consultation de dictionnaire. A la sortie, nous obtiendrons, en plus et indépendamment du texte imprimé, une préparation du volume entier, qui nous informera, phrase par phrase, sur les difficultés de traduction vers une ou plusieurs langues et la manière de les surmonter.

Mais avant que nous ne soyons en possession de cet immense fichier et d'une réponse instantanée à toutes nos questions sur un écran de télévision les terminologues doivent répondre aux besoins journaliers par des moyens plus classiques. En aucun cas leurs obligations courantes ne peuvent être diminuées ni ralenties sous prétexte de construction pour le futur. Par ailleurs une réalisation à longue échéance ne peut être effectuée sérieusement en marge de la connaissance des besoins pratiques actuels. C'est pourquoi nous nous servons de leur propre fichier, pour notre dictionnaire automatique, mais aussi pour l'édition de glossaires phraséologiques à l'aide d'un programme qui évite toutes les manipulations ordinaires de la typographie.

Ce que le terminologue introduit dans la machine, pour construire le dictionnaire c'est le contenu d'une fiche, c'est-à-dire un fragment de texte plus ou moins long (jusqu'à vingt mots environ) ainsi qu'un code-matière et une référence bibliographique.

in the brain).

The first task of a CTP is to decide on any re-ordering needed. It implements such a decision simply by rearranging the addresses in the element concerned. Each CTP entered does this, so that the individual items are met in their new order and can be added to the output string at once.

The selection and inflection of equivalents are carried out at the time they are to be produced, when all relevant information is available to the CTP without excursions into other parts of the structure. The insertion of function words, on the other hand, may be done by any CTP.

The resulting English output string is then passed to a final program which is responsible for format control. The normal form of output is punched paper tape, from which the printed copy, as shown in McDaniel et al.(this conference), is produced on a 'Flexowriter'. There is an alternative form of output on punched cards, from which printed copy can be produced on a card-controlled typewriter. This earlier form gives the text in the two languages side by side, which was useful for research purposes, but the absence of lower-case Roman letters and pagination, and the restricted width of each language version, makes this form less well suited for general use.

This format control process, and the main control routine which deals with the exploration of the tree and the handover from one CTP to the next, need not be described further, but the tasks of the individual CTPs will be outlined below.

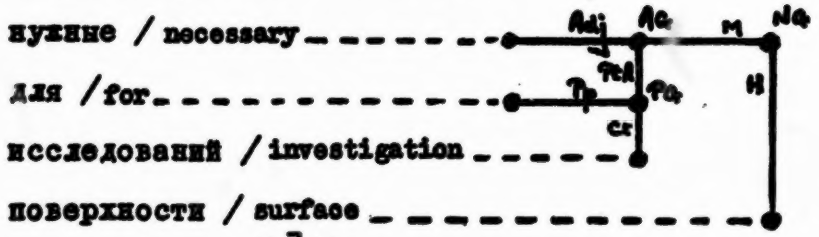
Tasks of nominal group CTP

(1) To insert before the group a preposition depending on the case and role of the group, e.g. of is inserted if case is genitive and role is qualifier in NG. Several instances occur in the sample output referred to above.

(2) To move modifiers containing items after the adjective or participle to the end of the group, with appropriate punctuation.

Example:

Structure as received from analysis:

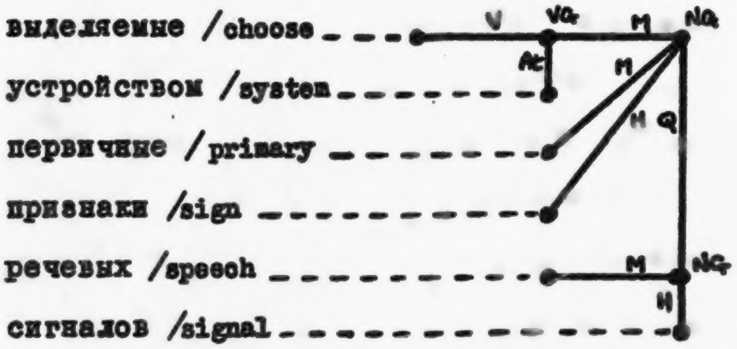


Result:

surfaces necessary for investigations

In a more complex case commas are inserted.

Example:


выделяемые /choose
устройством /system
первичные /primary
признаки /sign
речевых /speech
сигналов /signal
V
VG
M
NG
At
M
H
Q
M
NG
H

Result:

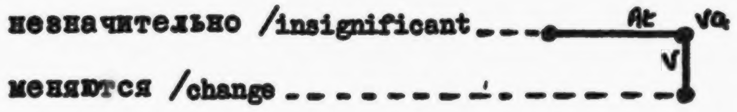
primary signs of speech signals, chosen by system,----

Tasks of verbal group CTP

(1) To insert auxiliary verbs and 'not' as necessary in finite verbal groups, for instance inserting does not for the present tense 3rd person singular negative. The precise rules for the position of the insertion are complex, but roughly these words are inserted immediately before the verb in negative verbal groups and before the verb and any immediately preceding adverbs in positive verbal groups.

Example:

Structure as received from analysis:


незначительно /insignificant
меняются /change
At
VG
V

The VG has systems coding 3rd plur., present, ся passive positive. The VG CTP therefore outputs are and hands control to the adjective CTP (since the dictionary entry for the first word is an adjectival one). As described below, this CTP will output the adjectival equivalent with an adverbial inflection -ly. The verb CTP then generates the verbal equivalent again with the appropriate inflection.

Result:

are insignificantly changed

(2) The VG CTP also inserts auxiliary verbs before "short form" predicative adjectives and participles, and inserts to before infinitives, in both cases with appropriate placing of not and any adverbs.

(3) Special measures are taken to allow for the non-standard behaviour (as regards English auxiliaries) when equivalents include be, should or can.

(4) The CTP is so arranged that a treatment of government phenomena could be added conveniently. The routine concerned was developed only as far as the flowchart stage.

Tasks of clause CTP

The principal task of this CTP is to determine the order of subject, verb and complements. For example, if in Russian a sentence begins with an intransitive verb, and the subject follows, the preferred translation depends on the length of the subject-short subjects can be put before the verb, but with long subjects this would not be acceptable in English and some expedient, such as the insertion of the dummy subject there, must be adopted (e.g. Then there arose the problem of ....).

Unlike the other CTPs described, this one was not implemented, being developed only as far as the flowchart stage. In its absence, certain pronominal subjects are inserted by ad hoc methods.

Tasks of noun, verb and adjective CTPs

Apart from certain insertions (such as having before past verbal adverbs), the main task of these CTPs is inflection. The decision to inflect is based on the systems coding and, in the case of adverb formation, on the role given to the item by the analysis. The actual type of inflection is chosen according to a code in the dictionary associated with each correspondent; thus boundary will be pluralised as boundaries , foot as feet , and so on. (Irregular forms such as feet are extracted by the program from a list, using an address given in the dictionary entry. Including both nouns and verbs, this list contains 212 forms). Provision is made for inflecting the right word in multiple word correspondents such as mode of life . All vagaries of English inflection called for by present dictionary equivalents are covered.

Selection of equivalents

There are five CTPs which select equivalents on various grammatical criteria, usually the role of the item. A typical case is that dealing with 'noun/adjectives' such as другой This ensures that другими авторами is translated by other authors, while границы другой is translated as boundary(s) of another (assuming, of course, that the analysis has given them structures of modifier-head and head-qualifier respectively).

Conclusion

The model and synthesis algorithm described proved satisfactory in practical use. They have the advantage that translations can be produced when the algorithms are incomplete: provided the sub-trees produced by a partial analysis are linked arbitrarily to produce a single sentence structure, this can then be explored by a synthesis algorithm, even one in which several CTPs are replaced by dummies. As new packages (analysis passes or synthesis CTPs) become available they can be incorporated very simply.

The work described above has been carried out at the National Physical Laboratory.

References

1. Halliday, M.A.K. (1961) - Categories of the theory of grammar. Word, 17, (3), pp. 241-292.
2. McDaniel, J. et al. (1967) - An evaluation of the usefulness of machine translations produced at Teddington, and an account of the translation methods. (This conference).
3. Yngve, V.H. (1960) - A model and an hypothesis for language structure. Proc. Am. Phil. Soc., 104, (5), pp. 444-466.

22

UN MODELE DE LA NOTION DU CAS

by A. ZALIZNIAK
Institut d'Etudes Slaves de l'Académie des Sciences
Groupe Structural
30 a avenue Troubnikowskĭ
MOSCOU

RESUME

On appelle segment toute partie d'un texte écrit délimitée par deux espaces en blanc voisins. Admettons que nous savons déterminer si une suite donnée de segments forme une phrase grammaticalement correcte. Admettons que nous avons relevé de quelque manière tous les segments appartenant aux substantifs (= "segments substantivaux"). Admettons enfin que nous savons déterminer si les deux segments substantivaux donnés appartiennent au même substantif pris au même nombre grammatical (autrement dit, s'ils désignent le même objet).

On appelle contexte une suite d'un nombre fini de segments et d'un symbole "...". On dit qu'un contexte A admet un segment m si l'on obtient une phrase grammaticalement correcte en remplaçant dans le contexte A, le symbole "..." par le segment m.

L'ensemble de tous les segments qu'admet un contexte A s'appelle colonne de ce contexte, et l'ensemble de tous les segments substantivaux qu'il admet, sa sous-colonne substantivale. P.ex. en russe, la colonne du contexte "вижу..." est сад, стену, вас, глазами, хорошо, терерь, етс etc. ; sa sous-colonne substantivale est сад, стену, вас, глазами...(mais sans хорошо,теперь, etc.). On appelle colonne, ou sous-colonne substantivale (en général) tout ensemble de segments qui sert de colonne (de sous-colonne substantivale) à quelque contexte.

Une sous-colonne substantivale est dite complète si chaque substantif pris à chaque nombre y est représenté par un segment au moins. P.ex. la sous-colonne substantivale du contexte вижу ... est complète, tandis que celle du contexte вижу большую... ne l'est pas (puisque p.ex. les substantifs au pluriel n'y sont pas représentés).

Une sous-colonne substantivale complète qui ne contient pas d'autre

sous-colonne substantivale complète (différente d'elle-même) s'appelle série casuelle (ou série à ca identique).

Au point de vue de la grammaire ordinaire, chaque série casuelle englobe l'ensemble de tous les segments substantivaux susceptibles d'avoir la signification grammaticale d'un cas déterminé. Autrement dit, chaque série casuelle correspond à un cas déterminé.

Le modèle décrit pourrait se heurter à certaines difficultés dans les langues à plusieurs cas locatifs (p. ex. dans les langues finno-ougriennes). Toutefois, les emplois secondaires de ces cas locatifs (p. ex. à titre de complément de certains verbes), qui existent presque toujours, doivent permettre de distinguer ces cas dans le cadre de ce modèle.

Une difficulté particulière, et assez rare, est représentée par ce qu'on appelle "le second génitif" du russe (à valeur partitive : чаю "du thé", сахару "du sucre", etc.). En russe moderne, toute forme du partitif (p.ex чаю) peut toujours être remplacée par un simple génitif (p.ex. чая) sans changement de sens. Par conséquent, le second génitif ne figurera pas sur la liste des cas russes distingués par le présent modèle (cette liste contenant le nominatif, le génitif, le datif, l'accusatif, l'instrumental, le prépositionnel et "le second prépositionnel"). Pour faire figurer le second génitif sur la liste des cas, le modèle doit être rendu considérablement plus compliqué (en particulier, il faudra recourir à une admission supplémentaire).

La particularité décrite du second génitif permet de distinguer les cas à expression formelle stable (p.ex. les sept cas du russe) et les cas à expression formelle instable (p.ex. le second génitif). Ce n'est qu'une question de terminologie si le nom de cas peut être accordé aux deux groupes ou s'il doit être réservé au premier seulement. Le modèle proposé est celui de la notion du cas à expression formelle stable.

23

Une troisième version du " Dicautom " .

J.A. Bachrach et L. Hirschberg.

Notre dictionnaire automatique fournit la traduction des termes d'un texte, commentée par des exemples, à des équipes de traducteurs et d'interprètes qui travaillent sur les quatre langues de la Communauté Européenne et sur l'anglais.

Il se différencie d'un dictionnaire, fait de papier et de carton, essentiellement par les moyens automatiques dont il dispose pour le tri, le stockage, la mise à jour, la consultation ou l'impression. Mais s'il est plus riche en exemples et manipulé en ordinateur, il a été réuni par des lexicologues, des terminologues et des traducteurs, comme un dictionnaire ordinaire.

L'équipe responsable des versions successives se composait des membres du Bureau de Terminologie de la Haute Autorité de la Ceca, J.A. Bachrach, R. Denti, P. Fleurbaay, J. Goetschaelckx, H. Lenoch, C. Noss, R. Weisgerber, J. Wilson, et du Centre de Linguistique Automatique Appliquée de l'Université libre de Bruxelles, J. Blois, Fr. Decresy, L. Delhaye, R. Dorrepaal-Van Beek, L. Hirschberg, N. Hollaar, J. Mommens, E. Morlet.

La description présentée ici concerne une troisième version du système, fonctionnant sur une IBM 360/40, achevée en décembre 1966 et mise au point au départ de la langue allemande. La première version sur IBM 1401 et au départ du français, date de 1962 (" La Traduction Automatique " - IV-3-1963). La deuxième a été présentée à Venise par J.A. Bachrach au " Séminaire International de l'OTAN sur la Documentation Automatique " (juillet 1963) et publiée depuis en divers endroits. Elle fonctionne sur IBM 1410, au départ du français et du néerlandais. Ainsi qu'il ressort du rapport " Language and Machine ", publié fin 1966 par la National Academy of Sciences, Washington, il n'existe qu'une seule formule qui se rapproche de la nôtre, à savoir celle adoptée par le service de traduction de l'armée allemande à Mannheim.

Au cours de cette nouvelle expérience nous avons précisé les définitions des concepts linguistiques, simplifié et systématisé les codages manuels. Nous avons réanalysé et transformé le processus et réorganisé les fichiers. Enfin, en tenant compte des possibilités nouvelles de la machine, d'une mémoire de 256K, et en fixant les modes d'exploitation des programmes, nous en avons considérablement amélioré le rendement.

Avant d'entrer dans le détail de la description du système, il est bon de donner un aperçu de son utilisation et des motivations qui ont guidé sa conception.

Notre dictionnaire comporte un mot ou une locution de la langue source, sa ou ses traductions dans les langues cibles, et pour chaque unité des exemples de contextes et leurs traductions. Ce qui est valable pour la langue générale, l'est également pour la langue technique ; la traduction d'un terme ne dépend pas seulement du domaine, de la matière traitée, de l'époque où il a été employé, mais aussi des contextes dans lesquels il se place. Toutes ces données sont réunies dans le dictionnaire.
Quelle sera la source de nos exemples ? Les terminologues puisent de préférence leurs informations dans des documents originaux: manuels, rapports, catalogues, brochures scientifiques ou publicitaires et surtout dans les revues spécialisées. L'emploi de textes traduits en plusieurs langues serait plus simple, mais risquerait d'entrainer la confection d'un langage artificiel et de perpétuer des erreurs. Il importe de se servir dans chaque langue de la terminologie propre au secteur en question dans le pays d'origine de cette langue.

Le terminologue reçoit du traducteur des questions, sous forme de phrases-demandes ou de fragments de phrases dans lesquelles le traducteur a senti des pièges de traduction. Du stock de phrases-exemples que nous avons accumulées dans notre fichier - qui s'élève à quelque six mille unités aujourd'hui, mais peut arriver à cent mille - l'ordinateur doit extraire les exemples de traduction les "plus proches possibles" de ceux de la demande. La présence des contextes dans les réponses permet au traducteur de choisir lui-même la traduction des mots, en contrôlant le bien-fondé des traductions proposées par les exemples, relativement aux cas qui l'occupent. S'il y a doute, plusieurs réponses sortiront, parmi lesquelles le traducteur éliminera lui-même les réponses inadéquates. (Illustration I)

Une autre condition importante pour l'utilisateur est que les questions puissent être formulées librement. En aucun cas, la formulation de la question ne nécessite une connaissance même partielle de la réponses: tout l'effort de construction du système incombe au réalisateur. L'utilisateur souligne simplement un fragment de quelques mots. Pour que ce fragment de texte à traduire puisse être découpé sans précautions particulières, les terminologues doivent veiller à opérer un découpage judicieux pour obtenir les phrases-exemples.

L'absence de tout codage au moment de la demande, implique un codage effectué lors de la construction du dictionnaire, qui donne des lois d'identification des formes du texte courant avec les unités du lexique. Une telle structure se justifie par le fait que le programme est conçu pour consulter, fréquemment et en vue d'une très grande quantité de textes à traduire, un dictionnaire de phrases relativement limité, de sorte que le codage à l'entrée est le plus économique. (Illustration II)

Illustration I

demande 0000000012

eine § traenke dieser art besteht aus einem robusten verzinkten § stahlblechgehaeuse

réponses

traenke	als automatische traenken haben sich tropftraenken und rinnentraenken bewaehrt fra gli abbeveratoi automatici si sono affermati quelli a goccia e quelli a livello costante
bestehen aus blech	die verteilerkoepfe der melkzeuge bestehen aus gedrueckten nichtrostenden stahlblechteilen i distributori delle unita di mungitura sono composti di pezzi a rilievo in lamiera di acciaio inossidabile
bestehen aus stahl blech	luftheizungsoefen bestehen aus einem inneren ofenmantel aus stahl und einem aeusseren stahlblechmantel le stufe ad aria calda sono costituite di un rivestimento del forno in acciaio e di un rivestimento metallico esterno
stahl blech verzinken	selbsttragende pfannen aus verzinktem stahlblech elementi autoportanti in lamiera d acciaio zincata
blech gehaeuse	ventilatoren fuer unterdachtrocknung werden zum ortsfesten einbau in stahlblechgehaeuse montiert geliefert i ventilatori per l essiccamento dell erba in fienile sono disponibili con involucri in lamiera d acciaio per montaggio stabile nel fienile

demande 0000000013

hochfoerderrinne mit schubstangen und § sperrechen

pas de réponse.

Illustration II

T1PV	SOLIVE	SOLIVES-HOURDIS
T1PV	SOLLICITATION	SOLLICITATION
T1PV	SOLLICITATION	SOLLICITATIONS
T3PF	SOLLICITE	SOLLICITES
T1PM	SONDAGE	SONDAGE
T1PV	SORTE	SORTE
T1PV	SOUDABILITE	SOUDABILITE
T3PI	SOUDABLE	SOUDABLE
T1PM	SOUDAGE	SOUDAGE
T9	SOUDER	SOUDE
T9	SOUDER	SOUDEE
T9	SOUDER	SOUDER
T9	SOUDER	SOUDES
T3PE	SOUDEUR	SOUDEUR
T3PE	SOUDEUR	SOUDEUSE
T1PV	SOUDURE	SOUDURE
T1PV	SOUDURE	SOUDURES
T3PF	SOUFFLE	SOUFFLEE
T1PV	SOUFFLERIE	SOUFFLERIE
T1PM	SOUFRE	SOUFRE
T1PV	SOURCE	SOURCE
T8	SOUS	SOUS-FACE
T8	SOUS	SOUS-GROUPES
T8	SOUS	SOUS-SOL
T3PF	SOUTERRAIN	SOUTERRAINES
T1PM	SOUTIEN	SOUTIEN
T3XF	SPATIAL	SPATIALE
T3XF	SPECIAL	SPECIAL
T3XF	SPECIAL	SPECIALE
T3XF	SPECIAL	SPECIAUX
T1PV	SPECIFICATION	SPECIFICATION

Ce fragment contient une difficulté de traduction et le contexte qui conditionne l'emploi de cette tournure particulière. Si nous avons en français, " PONT A POUTRES EN TREILLIS " , le même énoncé est repris dans cinq ou six langues sur la même fiche. Il servira de"phrase-exemple" (au singulier) qui ira dans le fichier.

F PONT A POUTRES EN TREILLIS

D FACHWERKBALKENBRUECKE

I PONTE A TRAVI A TRALICCIO (Illustration III)

N VAKWERKLIGGERBRUG

E TRUSSED GIRDER BRIDGE

Le premier travail de la machine est effectué par un programme du genre "KWIC" qui attache cette fiche toute entière, avec les énoncés, dans toutes les langues, en autant d'endroits différents qu'elle contient de mots: pont, poutres, treillis, vakwerkliggerbrug, bridge etc. En chaque endroit, la fiche va rejoindre d'autres fiches conçues comme elle. (Illustration IV)

En somme, le même stock de phrases-exemples, existant actuellement en cinq langues, est utilisé pour cinq dictionnaires, un dictionnaire par langue de départ. Toutefois, le programme est prévu pour six langues. Il s'ensuit que cinq réponses peuvent être données à partir d'une langue, vers cinq langues différentes. Cela équivaut finalement à trente dictionnaires bilingues.

Insistons maintenant sur le fait que les listes-dictionnaires finales contiennent des "unités lexicales" et non pas des mots, car ces unités sont parfois plus longues, comme "plaque de façade" ou "sauerstoffgeblasenes Eisen", tenus pour des unités. Parfois elles sont plus courtes. Ainsi il faut trois unités lexicales "Rost" "Stab" et "Eisen" pour reconstituer automatiquement c'est-à-dire reconnaître le mot "Roststabeisen" dans le texte d'une demande.

C'est le constructeur qui choisit l'unité lexicale et marque pour chacune d'elles les phrases du stock qui la concernent. Il emploie des critères d'observation basés sur les phrases-exemples et leurs traductions.

Lorsqu'un mot se traduit de plusieurs manières différentes, notre système donne au lexicologue le choix entre deux solutions. Ou bien il crée pour le mot plusieurs rubriques, chaque rubrique contenant d'autres phrases-exemples, et nous disons qu'il s'agit d'autant d' "homonymes" différents de ce mot. Ainsi en allemand "Raum" se traduira dans de nombreux contextes par "espace", dans d'autres par "chambre" ou "local" et enfin nous lui avons même trouvé la traduction "surface". Nous créons des "homonymes", lorsque le nombre de traductions n'est pas trop grand, et les contextes où ces traductions s'imposent sont nombreux.

Au contraire, nous avons trouvé pour "Lager" plusieurs traductions très particulières qui ne convenaient chacune qu'à un contexte bien défini. Pour définir les cas où il fallait choisir ces traductions, le mieux était de faire des "groupes locutionnels" de ces contextes, c'est-à-dire des unités indépendantes composées de plusieurs autres unités. Notre lexique allemand contient de cette manière "Kugellager" (roulement à bille), "Gleitlager" (appui glissant), "Obstlagerhaus" (dépôt ou entrepôt de fruits), "Lagerbehaelter" (cellule de stockage).

Illustration IV

PONT
PARAPETS DE ROUTES ET DE PONTS
D STRASSEN- UND BRUECKENGELAENDER
I PARAPETTI STRADALI E DI PONTI
N BORSTWERINGEN VOOR WEGEN EN BRUGGEN
E ROAD AND BRIDGE RAILINGS

PONT
PONT A POUTRES EN TREILLIS ET ARTICULATIONS INTERMEDIAIRES
D FACHWERKBALKENBRUECKE MIT ZWISCHENGELENKEN
I PONTE A TRAVI A TRALICCIO ARTICOLAZIONI INTERMEDIE
N VAKWERKLIGGERBRUGGEN MET SCHARNIERENDE DIAGONALEN
E LATTICE GIRDER BRIDGES WITH INTERMEDIATE JOINTS

PONT
PONT A POUTRES EN TREILLIS
D FACHWERKBALKENBRUECKE
I PONTE A TRAVI A TRALICCIO
N VAKWERKLIGGERBRUG
E TRUSSED GIRDER BRIDGE

PONT
PONT BASCULANT
D KLAPPBRUECKE
I PONTE LEVATOIO.,PONTE RIBALTABILE
N KLAPBRUG.,OPHAALBRUG.,BASCULEBRUG
E BASCULE BRIDGE

PONT
PONT CANTILEVER EN TREILLIS
D FACHWERK-GERBERTRAEGER-BRUECKE
I PONTE A SBALZO A TRALICCIO
N VAKWERK-GERBERLIGGERBRUG
E CANTILEVER TRUSS BRIDGE

Les "groupes locutionnels", comme les autres unités lexicales, possèdent dans notre dictionnaire des phrases-exemples qui leur sont attachées. Mais il s'agit de reconnaître de telles unités dans le texte d'une demande.

C'est donc par une analyse morphologique au sens le plus large de ce terme, allant de la reconnaissance d'une forme et de sa découpe à la découverte d'un groupement de plusieurs formes dans une phrase, que commencent les programmes de consultation automatique de nos dictionnaires. Ils comportent deux phases importantes en dehors de routines de tri et de mises en page, la deuxième grande étape étant le choix des meilleures phrases-exemples à donner en réponse à une certaine demande. Ce choix est fait parmi les phrases attachées aux rubriques du dictionnaire qui viennent d'être reconnues par l'analyse morphologique.

Les phrases-réponses les meilleures sont choisies par programme. Nous avons élaboré dans ce but des règles précises et un calcul assez compliqué. Mais en un mot il est permis de dire que ce sont les phrases parmi celles du stock qui ont en elles chacune le plus grand nombre possible d'unités lexicales reconnues dans une demande par la machine.

Le dictionnaire se compose de quatre fichiers indépendants, ce qui accélère la consultation automatique et facilite la correction éventuelle de ces fichiers.

Le premier est un dictionnaire alphabétique de mots ou de radicaux, accompagnés de leurs codes morphologiques et de deux numéros de renvoi, le numéro de la liste des groupes locutionnels auxquels ils appartiennent et le numéro de la liste des phrases-exemples dans lesquelles ils apparaissent comme mots simples. L'une des deux listes peut d'ailleurs ne pas exister.

Ce premier fichier est suffisamment dépouillé pour qu'il puisse être chargé en mémoire rapide au moment de la consultation du dictionnaire. Une table d'adresses constituée par les deux premières lettres des mots se forme au même moment dans cette mémoire. Cela permet une analyse morphologique très rapide, sans même trier alphabétiquement les mots d'une demande. (Illustration V)

Toutes les hypothèses de découpes sont faites pour chaque mot, de gauche à droite. Le programme tient compte aussi de l'existence possible, de désinences finales à la fin d'une séquence de lettres.

De plus il suit des règles d'acception et de rejet des diverses découpes, dont nous ne donnerons qu'un exemple d'application. Ainsi on voit dans l'illustration V, que " SERIEN|BAUTEN " a été découpée en deux mots, tandis que la découpe "BAU|KASTEN|SYSTEM" a été rejetée (un signe "moins" se trouve devant le sigle DE) au profit de la découpe "BAUKASTEN|SYSTEM". Il est évident qu'il refuse aussi toute hypothèse qui ne recouvre pas convenablement une séquence de lettres sans blancs.

Illustration V

```
MCT
MCT  VORGEFERTIGTE
 DE
   F  VOR
   F  GEFERTIGTE
MOT  STANDARDBAUTEN
 DE
   F  STANDARD
   F  BAUTEN
 DE
   F  STANDARD
   F  BAUTEN
 DE
   F  STANDARD
   F  BAUTEN
MCT  UND
MOT  SERIENBAUTEN
 DE
   F  SERIEN
   F  BAUTEN
MOT  IM
 DE
   F  IM
MOT  BAUKASTENSYSTEM
-DE
   F  BAU
   F  KASTEN
   F  SYSTEM
-DE
   F  BAU
   F  KASTEN
   F  SYSTEM
-DE
   F  BAU
   F  KASTEN
   F  SYSTEM
 DE
   F  BAUKASTEN
   F  SYSTEM
MCT  •
MCT
0001  VOR
0002  FERTIG
0003  STANDARC
0004  BAU
0005  BAUTEN
0006  SERIE
0007  IM
0008  BAUKASTEN
0009  SYSTEM
TEMPS  EN  SEC= 000003.6
```

```
00000000  0000C5D8  00000
0000C5A0  0000C6DC  0000C
00000000  00000000  00000
00000000  0000C658  0000C
0000C618  00000000  0000C
0000C5D8  0000C874  0000C
00000000  0000C7B4  00000
00000000  0000C744  0000C
0000C71C  00000000  0000C
0000C708  0000C814  00000
00000000  0000C7F0  0000C
0000C7C8  00000000  0000C
0000C7B4  00000000  00000
00000000  0000C850  0000C
0000C828  00000000  0000C
0000C6DC  0000C894  0000C
0000C874  0000C990  0000C
00000000  00000000  00000
00000000  0000C918  0000C
0000C8D4  00000000  0000C
0000C894  0000C9E4  0000C
00000000  00000000  00000
00000000  00000000  0000C
0000C990  0000CBE4  0000C
00000000  0000CA8C  00000
00000000  0000CA44  0000C
0000CA24  0000CA68  0000C
0000CA44  00000000  0000C
0000CA10  0000CB08  00000
00000000  0000CAC0  0000C
0000CAA0  0000CAE4  0000C
0000CAC0  00000000  0000C
0000CA8C  0000CB84  00000
00000000  0000CB3C  0000C
0000CB1C  0000CB60  0000C
0000CB3C  00000000  0000C
0000CB08  00000000  00000
00000000  0000CBC0  0000C
0000CB98  00000000  0000C
0000C9E4  0000C5BC  0000C
0000CBE4  00000000  00000
```

Tout le processus n'a duré que 3,6 secondes pour une demande de six mots dont quatre ont été découpés en deux.

Le numéro de liste des groupes locutionnels est le premier argument de tri du deuxième fichier, qui donne ainsi pour chaque mot les numéros des groupes locutionnels auxquels il appartient dans le dictionnaire. Les mots eux-mêmes ne réapparaissent pas dans cette deuxième liste, mais seulement leurs caractéristiques relativement à chaque groupe, qui permettent de vérifier si les conditions d'existence du groupe sont remplies dans un texte que l'on analyse. (Illustration VI)

Après la consultation des deux premiers fichiers, les mots d'une demande ont été découpés en fragments ou (et) réunis en groupes locutionnels. Chaque fragment peut avoir aussi été identifié avec plusieurs homonymes.

A chaque homonyme libre, comme à chaque fragment d'un groupe locutionnel, correspond une liste des numéros des phrases-réponses valables pour lui. C'est le troisième fichier qui contient ces listes de numéros de phrases, sans contenir les phrases elles-mêmes. Il mentionne aussi les langues-cibles pour lesquelles ces phrases existent. On remarquera le gain de temps obtenu en effectuant l'analyse jusqu'ici sans se soucier de la langue cible. Ce n'est qu'au moment de la consultation du troisième fichier pour la recherche de la meilleure phrase-réponse que se fait la séparation du processus par langues demandées.

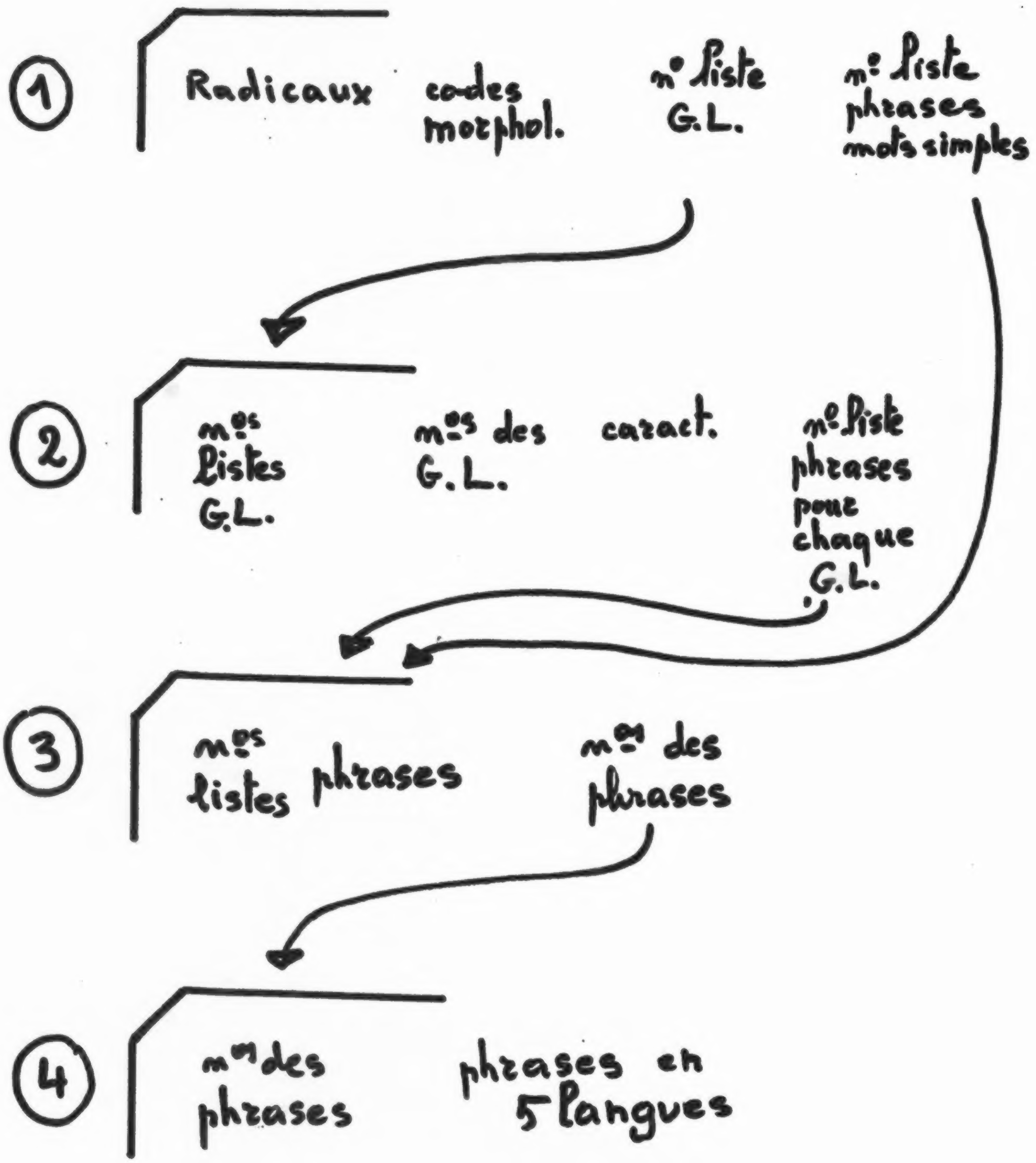
Après consultation de ce troisième fichier, le programme attache à chaque radical découpé dans une demande plusieurs paquets de numéros de phrases-réponses, chaque paquet correspondant à un homonyme ou à une hypothèse de groupe locutionnel. On obtient ainsi un tableau de numéros de phrases où chaque phrase est citée un certain nombre de fois. Nous attribuons à chaque phrase une cote le plus souvent égale à ce nombre, moyennant certaines précautions dans les cas de découpes multiples.

On choisit ensuite pour chaque fragment de la demande la phrase-réponse qui le concerne et qui possède la cote la plus élevée. Cela constitue en même temps un choix parmi ses homonymes. Cependant plusieurs homonymes peuvent être acceptés, avec la meilleure phrase-réponse de chacun d'eux, si elles ont toutes la même cote.

L'appartenance à un groupe locutionnel n'élimine pour un mot toutes ses autres hypothèses que si ce groupe a, dans le texte envisagé, une valeur de certitude qui est déterminée par programme. On se base sur des critères de certitude que nous ne pouvons énumérer ici. Disons par exemple qu'un verbe séparable n'est reconnu avec certitude que dans le cas où la particule est collée au radical verbal. Dans les autres cas on permet à plusieurs hypothèses d'entrer en compétition.

Ce n'est qu'au moment d'éditer les résultats que l'on retire du dernier fichier les phrases-exemples elles-mêmes, qui viennent d'être choisies.

Illustration VI


1
Radicaux
codes morphol.
n° liste G.L.
n° liste phrases mots simples
2
n°s listes G.L.
n°s des G.L.
caract.
n° liste phrases pour chaque G.L.
3
n°s listes phrases
n°s des phrases
4
n°s des phrases
phrases en 5 langues

Ce quatrième fichier, trié par numéros de phrases, peut être beaucoup plus volumineux que les autres et tenir, s'il le faut, dans une mémoire-mars.

Il nous reste à justifier quelques détails de l'analyse morphologique adoptée ici. Elle se caractérise tout d'abord par la comparaison des unités lexicales à partir de la gauche.
Les avantages de reconnaissance des formes à partir de la gauche sont liés au fait que la machine consulte d'abord le lexique des formes, et non pas des listes fermées de désinences, en commençant par la droite. Le premier avantage, sauf pour des unités lexicales de moins de quatre lettres et des préfixes, réside dans le fait que la machine va plus vite, parce qu'elle crée beaucoup moins d'hypothèses inutiles, ainsi que nous l'avons vérifié expérimentalement, en simulant les deux méthodes.

Mais il y a un deuxième avantage de loin le plus important surtout pour des langues comme l'allemand et le néerlandais qui agglomèrent en un mot des unités lexicales que l'on peut, -et que l'on doit, - séparer dans le lexique. En effet le programme peut effectuer en une seule fois les découpes des mots et des désinences, jusqu'à épuisement et sans retour en arrière.

Enfin, nous avons groupé les innombrables déclinaisons des substantifs allemands en admetant une seule liste de désinences pour tous, définie par le code signifiant substantif. (Illustration VII)
La simplification des déclinaisons allemandes au point de les ramener toutes à une seule, facilite énormément le codage du dictionnaire, si nous le comparons au codage nécessaire pour l'analyse morphologique élaborée par le groupe de Grenoble, qui tient compte de vingt-trois listes distinctes de désinences pour les substantifs allemands.

Que sacrifions-nous à cette simplification ?

Il n'y a aucun danger, dans le cas présent, à admettre pour un substantif des désinences qu'il ne prendra en réalité jamais. En effet si cela pouvait mener à confondre deux substantifs, il suffirait de donner toutes leurs formes dans le dictionnaire et de les traiter comme des mots invariables. Nous n'avons pas rencontré cette situation sur plusieurs milliers de mots de textes techniques traités ainsi. De toutes manières, dans le but d'obtenir une découpe rapide des composés, faite de gauche à droite et en un seul passage, nous donnons deux formes dans le dictionnaire pour les substantifs qui prennent un "umlaut" ou pour tout autre changement de radical.

Notre analyse perd une seule information, celle de savoir à quel cas et à quel nombre appartient un substantif donné dans un texte. Elle convient donc pour les applications pour lesquelles ces renseignements seraient superflus, telles que les consultations automatiques de dictionnaires qui ne seraient pas suivies d'analyses syntaxiques.

Illustration VII

mots qui les acceptent	noms		adjectifs		verbes			
					forts	faibles	participes	
désinences	F	I	FetI	F	F	F	F	I
blanc	○	○	○	○	○	○		○
CHEN	○							
CHENS	○							
E	○	○		○	○	○		○
EM				○				
EN	○	○		○	○	○		
END							○	
ER			○	○				
ES	○	○		○				
EST			○		○	○		
ET					○	○	○	
ETE						○		
ETEN						○		
ETEST						○		
ETET						○		
LEIN	○							
LEINS	○							
N	○	○			○	○		
ND							○	
NEN	○							
R				○				
S	○	○						
SE	○							
SEN	○							
SES	○							
ST					○	○		
T					○	○	○	
TE						○		
TEN						○		
TEST						○		
TET						○		

F = finales   I = intermédiaires.

Au moment de conclure, il est bon de passer en revue quelques points que soulève la lexicographie classique et de voir dans quelle mesure ils nous concernent ici.

Il s'agit principalement du choix des termes à introduire dans un dictionnaire, c'est-à-dire de la délimitation d'un domaine sémantique et du relevé du plus grand nombre possible de termes que s'y rapportent, ensuite des relations sémantiques entre les termes c'est-à-dire de la structuration sémantique à l'intérieur du lexique choisi; et enfin des problèmes d'adressage, c'est-à-dire de l'organisation matérielle en vue de la consultation, emplacement des termes, répétition éventuelle, orthographes multiples etc...

Si nous n'avons pas centré notre exposé sur ces questions, c'est que le traitement en ordinateur les résoud assez naturellement à condition d'introduire en machine les données complètes et primitives.

L'adressage tout d'abord est à ce point aisé que l'on peut multiplier à souhait les orthographes et que les problèmes d'emplacement et de répétition ne se posent plus.

En introduisant en machine des fragments de textes authentiques, nous notons en même temps leur référence bibliographique et un ou plusieurs codes matières définissant le contexte dont ils ont été extraits. Ces informations sont par là-même et sans autre travail attachées aux termes contenus dans ces fragments. Comme la mémoire de la machine est pratiquement illimitée, on ne doit refuser aucun terme à l'entrée, ni séparer explicitement des domaines, tels la sidérurgie, l'architecture etc... Tout au plus devra-t-on attacher à certaines unités lexicales un signe spécifiant qu'elles n'ont aucun intérêt par elles-mêmes dans un dictionnaire technique (par exemple pour des mots comme "faire", "grand", "froid" etc.).

Une liste parfois longue de codes-matières se trouve attachée à chaque unité par le truchement des diverses phrases-exemples. Il s'établit ainsi spontanément une structuration sémantique complexe à l'intérieur du système que l'on pourrait découvrir par des techniques taxonomiques. Elle est plus riche que la structure en arborescence (analogue à la classification décimale universelle)la seule qui soit réalisable dans un dictionnaire habituel, à côté du classement alphabétique.

Au moment de formuler une demande, sous forme de "phrase difficile à traduire", le champ sémantique est délimité implicitement par le fait que la machine ne cherche que des phrases réponses aussi proches que possible de la demande. Le traducteur reçoit un glossaire qui ne se rapporte qu'à son texte et dans l'ordre d'apparition des termes.

Une mise à jour régulière est prévue qui supprime le problème d'une certaine complétude au départ, et permet d'enregistrer rapidement ces néologismes qui font défaut dans les dictionnaires ordinaires, souvent dépassés dès leur sortie de presse ! Les pertes de temps considérables, occasionnées par des recherches fastidieuses et souvent vaines, seront réduites au strict minimum.

L'établissement d'une véritable collaboration entre les compilateurs du dictionnaire et les utilisateurs mettra à la disposition de tous les données dont disposent quelques-uns et qui risquent, autrement d'être enfouis à tout jamais dans quelque obscur fichier personnel.

Cet instrument souple, puisqu'adaptable à toutes sortes de demandes, peut non seulement faciliter considérablement le travail de son utilisateur, mais lui offre aussi, sans effort supplémentaire de sa part, la possibilité d'un travail plus rapide et de meilleure qualité, ainsi qu'une spécialisation sans peine.

Système maniable, facile à étendre et à corriger, l'outil de travail tel que nous le proposons offre de multiples avantages, et de loin de favoriser l'élimination du traducteur humain, il pourra bien au contraire constituer une aide appréciable dans son travail.

Notes bibliographiques.

I.A. MELCUK . Analyse morphologique en traduction automatique. "Problèmes de Cybermétique" Vol. 6 - 1961 , pp.207-276.

L.E. DOSTERT , R.R. MACDONALD . Georgetown University Machine Translation Research project . General report 1952 - 1963 (n°30) june 1963.

G. LAURENT , A. BERNARD-GEORGES , D. LEVENBACH . Etude Morphologique du substantif allemand -Grenoble, avril 1962 - EUR 525f. 1964.

J. BLOIS , Fr. DECRESY , J. MOMMENS . Analyse morphologique automatique du français . Université libre de Bruxelles . Mars 1963.

DICAUTOM . - La " Traduction Automatique " - 1963 , Vol.IV n°3 (Mouton)

L. HIRSCHBERG . Dictionnaires automatiques pour traducteurs humains. " Journal des Traducteurs ", Montréal , Vol.10, n°3 (1965)pp. 78-86.

J.A. BACHRACH . Une méthode en terminologie . Comptes rendus du " Colloque de Linguistique Appliquée " -Ceca Luxembourg- avril 1966.

L. HIRSCHBERG . Conception, construction et utilisation de dictionnaires automatiques multilingues. Comptes rendus du " Colloque de Linguistique Appliquée " - Ceca, Luxembourg - avril 1966.

25

Experimental Design.

Our procedure in setting up an automatic method for identifying personal names and titles was approximately as follows:

(1) We investigated permissible patterns of personal titles and of English, French, Russian, German, Spanish, Chinese, Arabic, and other personal names whose occurrence in texts we could anticipate.

(2) We obtained a 60,000-word sample of newspaper texts and determined: (a) patterns of occurrence of personal names and titles in texts, (b) patterns of personal names and titles occurring in texts, and (c) problems involved in distinguishing personal names and titles from each other and from other parts of texts.

(3) Based on (1) and (2) above, we set up an automatic procedure designed to identify personal names and titles in newspaper texts. This procedure was embodied in flowcharts and a dictionary of about 8,000 entries..

(4) We tested our procedure manually on a 100,000-word sample of new newspaper texts, and we amended the rules and expanded the dictionary on the basis of the information provided by the tests.

(5) We then stabilized the improved procedure and (a) tested it out manually on a new 40,000-word sample of newspaper texts and (b) collected statistics concerning its accuracy and exhaustiveness. (Our reasons for applying the algorithm manually were as follows: (a) our identification system was embodied in dictionary entries and flow charts which were sufficiently detailed to permit accurate execution of recognition procedures, and (b) we thought that it would not pay to code and debug over a period of months what would probably turn out to be a "one-shot" program.)

(6) We then investigated what types of errors had occurred and proposed various amendments to the automatic recognition procedure.

Some Problems of Automatic Identification of Personal Titles and Names.

Automatic identification of titles and names in texts is of course not without its difficulties. First of all, many personal names are orthographically identical with other types of words in the language. This is the case since among the main sources of surnames are: (1) titles (e.g., "King"), (2) names of occupations (e.g., "Baker"), (3) topographic terms (e.g., "Hill"), (4) personal attributes (e.g., "Coward"), (5) place names (e.g., "London"), (6) names of animals (e.g., "Fox"), (7) names of trees (e.g., "Pine"), etc.

There is considerable ambiguity between personal names and place names due to the fact that not only are the names of

AN EXPERIMENTAL SYSTEM FOR AUTOMATIC RECOGNITION OF PERSONAL TITLES AND PERSONAL NAMES IN NEWSPAPER TEXTS

Casimir Borkowski

Thomas J. Watson IBM Research Center

Yorktown Heights, N.Y.

Summary.

Natural language seems to contain various special-purpose sublanguages (e.g., personal titles, personal names) -- each with its own structure which relative to the total structure of language is quite simple.

An ability to generate and to recognize automatically words and word strings belonging to various special-purpose sublanguages may prove to be very useful since they play an important role in indexing and in various systems for extracting and distributing information.

This paper (1) describes some of the main problems involved in automatic recognition of personal titles and names in newspaper texts, (2) outlines some rules of an algorithm designed to perform this task, (3) presents statistics concerning the algorithm's accuracy and exhaustiveness obtained in manual application of the algorithm to texts, (4) discusses and interprets some of the results, and (5) suggests some applications for computer programs capable of recognizing personal titles and names.

***

Motivation for the Experiment.

One of the major questions of the day is the extent to which a computer can be instructed to identify various parts of texts written in plain, ordinary language. In trying to answer this question, we set ourselves the preliminary limited objective of developing an automatic procedure for identifying personal titles and personal names in English-language texts.

Experimental Design.

Our procedure in setting up an automatic method for identifying personal names and titles was approximately as follows:

(1) We investigated permissible patterns of personal titles and of English, French, Russian, German, Spanish, Chinese, Arabic, and other personal names whose occurrence in texts we could anticipate.

(2) We obtained a 60,000-word sample of newspaper texts and determined: (a) patterns of occurrence of personal names and titles in texts, (b) patterns of personal names and titles occurring in texts, and (c) problems involved in distinguishing personal names and titles from each other and from other parts of texts.

(3) Based on (1) and (2) above, we set up an automatic procedure designed to identify personal names and titles in newspaper texts. This procedure was embodied in flowcharts and a dictionary of about 8,000 entries.

(4) We tested our procedure manually on a 100,000-word sample of new newspaper texts, and we amended the rules and expanded the dictionary on the basis of the information provided by the tests.

(5) We then stabilized the improved procedure and (a) tested it out manually on a new 40,000-word sample of newspaper texts and (b) collected statistics concerning its accuracy and exhaustiveness. (Our reasons for applying the algorithm manually were as follows: (a) our identification system was embodied in dictionary entries and flow charts which were sufficiently detailed to permit accurate execution of recognition procedures, and (b) we thought that it would not pay to code and debug over a period of months what would probably turn out to be a "one-shot" program.)

(6) We then investigated what types of errors had occurred and proposed various amendments to the automatic recognition procedure.

Some Problems of Automatic Identification of Personal Titles and Names.

Automatic identification of titles and names in texts is of course not without its difficulties. First of all, many personal names are orthographically identical with other types of words in the language. This is the case since among the main sources of surnames are: (1) titles (e.g., "King"), (2) names of occupations (e.g., "Baker"), (3) topographic terms (e.g., "Hill"), (4) personal attributes (e.g., "Coward"), (5) place names (e.g., "London"), (6) names of animals (e.g., "Fox"), (7) names of trees (e.g., "Pine"), etc.

There is considerable ambiguity between personal names and place names due to the fact that not only are the names of places a frequent source of personal names but also because many localities were named after people, as for example, Elizabeth, New Jersey andDallas, Texas. And to make matters worse, hotels, business firms, universities, etc. can be named after people and are often referred to by an abbreviated name which is that of a person (e.g., "He is staying at the Hilton", "He graduated from Stanford", "Ford was hit by a strike last week", "Two of them climbed the Everest"). As for personal names like "Helena Rubinstein" and "Max Factor", they designate persons as well as business firms, while "Philip Morris" is the name of a person, of a corporation, and of a brand of cigarettes.

Yet another difficulty arises in case of names of persons (e.g., "Madison") when they perform a naming function with regard to something, say an avenue, (e.g., "Madison Avenue"). Presumbly, it would be worthwhile to distinguish automatically references to persons from references to things named after persons.

Further difficulties in automatic recognition result from the co-occurrence in texts of names belonging to different name strings (e.g., "John Byron" as in "Estelle gave John Byron's Don Juan", "Mary Jane" as in "For Mary Jane had nothing but sympathy", "Alexander Montgomery" as in "According to Alexander Montgomery was slow in exploiting successes").

Other difficulties in recognizing personal names result from the fact that personal titles are not unfailing aids in identifying and disambiguating personal names since titles themselves can be homographic with other types of words. For instance, "General" is a military rank in "General Mobutu", but not in "General Motors".

Further difficulties result from the fact that some titles are homographic with given names. How is an automaton to tell that "Dean" is a title in "Dean Wiesner" but a name in "Dean Rusk", that "King" is a title in "King James" but a name in "James King", that "Earl" is a title in "the fourth Earl Russell" but a name in "the Chief Justice Earl Warren"?

Some Recognition Rules.

Our recognition algorithm was intended as a frame of reference in an investigation of the trade-off between the efficiency and the complexity of a series of algorithms.

To be able to investigate the trade-off between the efficiency and the complexity of a series of algorithms, we avoided taking as our point of departure a strong theory about the structure of language, semantics, pragmatics, etc., and about the amount of syntactic recognition required for successful identification in texts of personal titles and names. Instead, we sought to discover what assumptions and what information about natural language and texts may be pertinent to the resolution of the limited problems in recognition which we set for ourselves.

Stronger assumptions and more elaborate techniques of analysis can be built into subsequent algorithms if required and as required. For instance, since parsing may be helpful in identifying sequences of names each of which is followed by its title (e.g., "The President nominated John Gordon Ambassador to Guatemala, William T. M. Beale Jr. Ambassador to Jamaica...") future recognition algorithms may parse sentences containing: (1) double-object verbs (e.g.,"nominate") and (2) strings consisting of personal names followed by titles.

At a later time, parsing and/or other types of analyses may be extended to sentences, paragraphs, and articles containing other kinds of words and phrases. However, since parsing and other types of analyses may be expensive, it would seem advisable to apply them only when they can reasonably be expected to provide economic solutions to valid problems.

Our rules describe the arrangement in the sentence of the words, phrases, and punctuation marks which are pertinent to recognition of names and titles. Generally, the description starts with the first, that is, the leftmost pertinent element and terminates with the last, or rightmost pertinent element. Recognition rules were given a "left-to-right" format because rules expressed in this way are easy to implement on an electronic computer.

For greater ease of understanding, the rules are expressed here in narrative form. For the sake of brevity, only some recognition rules are listed here. A more complete description of identification rules is available elsewhere.(1)

Our rules for recognizing names of persons take advantage of the style rules of The New York Times. We would conjecture that whereas details of name recognition rules may vary from newspaper to newspaper, their general pattern will remain fairly stable and independent of editorial conventions.

The rule for identifying personal titles which was selected as a reasonable first approximation states that a word or phrase in text is a personal title either:

(1) if it matches a word or string of words on a list of titles

or

(2) if it matches a word or a string of words which is on a list of words and phrases which commonly combine with titles (e.g., "Acting", "Assistant", "Vice") and is followed by a personal title (e.g., "Acting Mayor", "Acting Assistant Vice President")

or

(3) if it is a personal title followed by a word or a string of words which is on a list of words which commonly combine with titles (e.g., "-elect", " at Large", "pro tempore") as in "Senator-elect", "Ambassador at Large", "President pro tempore".

or

(4) if it is a title designated by a list, like say "Commissioner", followed by the word "of" and any capitalized word (e.g., "Commissioner of Parks").

or

(5) if it is a word beginning with a capital letter and followed by a title designated by a list (e.g., "Commissioner" as in "Police Commissioner").

A preliminary (and a highly tentative) rule specifies how titles concatanate. This rule permits distinguishing some strings such as "Prime Minister, Sir" as in "Prime Minister, Sir Alec Douglas-Home", "Rev. Dr." as in "Rev. Dr. Martin Luther King", "Mr. Chairman"," Mr. Counsel", "Mr. Chairman, Ladies, and Gentlemen:", and so forth from titles followed by names.

The present set of rules for identifying titles which are homographic (that is, ortographically identical) with other words is relatively simple. It divides ambiguous titles into four classes; words of Class I (e.g., "King", "Pope", "Prince") are assumed to be titles (e.g., "King John", "Pope John") unless:
(1) preceded by either personal titles designated by a list such as "Mr.", "Dr.", "M. Sgt.", "General", etc., or by given names and initials in various combinations (e.g., "Mr. King", "John King", "Dr. Pope", "John Pope", "John M. King", "J. M. King")

or

(2) followed by such postnomial elements as "Sr." (e.g., "King, Sr."), "& Bros." (e.g., "King & Bros.), "and Company" (e.g., "King and Company"), and so forth.

Occasionally, words of Class I are followed by capitalized words or phrases which designate various institutions, establishments, locations, and so forth which are frequently named after persons (e.g., "Drug Store", "College", "Avenue", "Theorem"). Although, the list of such words and phrases is open-ended, its most frequently occurring members can be discovered and listed quite easily. Furthermore, there is some evidence that many or most such phrases can be identified by means of recognition rules. Words which are members of Class I are assumed to be names when they are followed by words such as "College" or phrases such as "Drug Store".

Words which are members of Class II (e.g., "Kaiser", "Chamberlain", "Earl") are assumed to be personal names. Commonly occurring exceptions to this rule (e.g., "Kaiser Wilhelm", "Lord Chamberlain", "Earl of" (if followed by a word beginning with a capital letter)) are listed.

Words which are members of Classes III and IV (e.g., "General", "Principal", "Justice") are assumed to be titles. Commonly occurring exceptions to this rule are listed ("General Assembly", "Major Medical Plan", "Principal Investigator", "Justice Department").

As our rules become more sophisticated, the need for lists of exceptions will diminish. However, it is likely that listing exceptions will often be an attractive alternative to rendering a rule more complicated.

Personal titles in the plural are recognized by means of a simple rule. It states that a string of characters is a personal title in the plural if:

(1) it is recognizable as a personal title

and if either

(2) its final word is followed by the letter "s" (e.g., "Major Generals")

or else

(3) if one of the words of which it is composed and which a list designates as the stem for the plural is followed by the letter "s" (e.g., "Collector" as in "District Collector of Internal Revenue").

A procedure similar to the one for identifying titles in the plural is used to recognize personal titles in the possessive case (e.g., singular: "Major General's", plural: "Major Generals'").

Our rules assume that the capitalized word or string of words and initials which frequently follows a title is the name of a person (e.g., "President Nkrumah", "Mr. Paul-Henri Spaak", "Governor Nelson Rockefeller").

If a title is followed by a word beginning with a lower-case letter or by certain punctuation marks, this indicates that the title is not followed by a name. However, occasionally personal titles are followed by names beginning with lower-case letters (e.g., "President de Gaulle").

On occasion, titles are followed by capitalized words which are not names. This happens in particular when a title is followed by a capitalized word or phrase which designates an institution, an establishment, a site, and so forth, named after a personal title (e.g., "Ambassador Bar", "Archduke Trio", "Emperor Concerto", "President Hotel", "Queens County", "Viceroy Lumber Company").

Although -- as mentioned earlier -- the list of such words and phrases is open-ended, its most frequently occurring members have been listed and are consequently identifiable. Commonly occurring exceptions to this rule are also listed and are therefore identifiable. In addition, we have some simple preliminary rules for identifying phrases whose designata often bear as names words or phrases which are personal titles (e.g., "President Radio Repair Shop", "Viceroy Lumber Company"). However, since the identification of phrases which designate such namesakes has been given little attention, these rules are very tentative.

Words which are generally names of weekdays when they occur after personal titles (e.g., "They saw the President Monday") are of course listable and therefore identifiable.

Occasionally, titles are followed by capitalized words which are not names and for the recognition of which the rules make no provisions (e.g., "British" in "The Prime Minister, British sources said, will arrive on Monday." and "New York" in "Mr. Stevenson prefers Washington and Mr. Rusk, the Secretary of State, New York."). Constructions such as these are, however, quite rare.

Occasionally, prepositional and other phrases intrude between a title and a name (e.g., "the French Ambassador to the United States, Herve Alphand", "the Foreign Minister of France, Maurice Couve de Murville"). Prepositional phrases of this sort and other adjuncts are identified by various rules of the "brute force" type whose statements are constructed as follows: If a personal title (e.g., "Foreign Minister") is followed by a phrase consisting of the preposition "of" and of the name of a country, such phrase is part of the title.

In general, in a string of words consisting of titles and names, titles precede names (e.g., "the Secretary of State, Dean Rusk, the Foreign Minister of the Federal Republic, Gerhard Schröder, the Foreign Minister of France, Maurice Couve de Murville"). Sequences of names each of which is followed by its title (e.g., "Dean Rusk, the Secretary of State, Gerhard Schröder, the Foreign Minister of the Federal Republic, ...") are rare. (Ordinarily, in a construction of this type, each title is set off from the name which follows it by a semi-colon.)

In spite of counter-examples such as the ones above, one can reasonably assume that if the capitalized word or string of words and initials which frequently follows a personal title is NOT an identifiable word like "Hotel", "Garage", "Street", "Monday", etc., or a phrase like "Barber Shop", "Drug Store", "Meat Packing Company", etc., then it is a personal name.

While counter-examples to this rule and to similar rules are easy to invent, the inventors of counter-examples usually miss the point that rules such as these are statistical observations and that in actual application to texts they hold up rather well. As stated earlier, among the goals of an investigation of this type is to obtain experimental evidence as to how well the rules hold up and what amendments are required to simplify them, to render them more accurate, and to expand their scope.

If a personal title (e.g., "President") is conjoined to the titles "Mrs." or "Miss" (as in "the President and Mrs. Johnson"), the capitalized word or string of words which follows the conjoined titles is generally the name of a person.

In general, the name of a person acts distributively with regard to the preceding titles, that is to say, a phrase like "the President and Mrs. Johnson" decomposes into "President Johnson and Mrs. Johnson".

Occasionally, a personal name does not act distributively with regard to conjoined titles which precede it (as in "an agreement between the Cardinal and Mrs. Johnson", "a meeting between the President and Mrs. Luce"); however, the present set of rules makes no provisions for recognizing such cases.

Generally, in newspaper articles, capitalized words and initials which (a) frequently follow a personal title in the plural (e.g., "Senators") and (b) which are not followed by other titles (e.g., "Senators, Congressmen, and Generals") are strings of personal names (e.g., "Senators Javits and Kennedy", "Senators Jacob Javits, Robert Kennedy and George D. Aiken", "Presidents Johnson and Lopez Mateos").

Of course, "Ambassadors Bar and Grill" is not a title in the plural followed by two names. However, commonly occurring phrases such as "Bar and Grill" are listable and therefore identifiable.

As a rule, the first name string -- which is often separated from the title by a comma -- terminates before the first conjunction "and" or before the next comma; the second name string begins after "and" or after the comma; etc.

If two name strings which follow a title in the plural (e.g., "Senators Jacob Javits, Robert Kennedy, ...") are separated by a comma, then -- generally -- the end of the second name string is marked by a comma or by the conjunction "and".

If two name strings which follow a title in the plural are separated by "and", then -- generally -- the end of the second name string is marked either by punctuation marks such as sentence period, a colon, a semi-colon, etc., or by a word beginning with a lower-case letter (e.g., "arrived" in "Presidents Johnson and Lopez Mateos arrived today."). However, if the word beginning with a lower-case letter is a name conjunction (e.g., "de", "von"), then such word does not mark the end of the second name (e.g., "Presidents Lyndon Johnson and Charles de Gaulle").

Occasionally, the string of words which follows a title in the plural consists of both names and prepositional phrases (e.g., "Senators Javits of New York and Fulbright of Arkansas", "Senators from New York, Javits and Kennedy"). Our rules permit identifying some prepositional and other phrases which may intrude between titles in the plural and the names which follow them.

Generally, the title in the plural acts distributively with regard to the names which follow it, that is to say, a phrase like "Senators Javits and Kennedy" decomposes into "Senator Javits and Senator Kennedy".

Generally, the end of a name string which may follow a personal title in the singular is marked by punctuation (comma, sentence period, dash, semi-colon, colon, exclamation point, apostrophe, three dots, left or right parenthesis, etc.) or by a word beginning with a lower-case letter.

However, a lower-case letter does not mark the end of a name string if the word which begins with it is either:

(1) a name conjunction (e.g., "de" as in "Attorney General Nicolas deB. Katzenbach")

or

(2) the last element of a hyphenated Chinese given name (e.g., "lai" in "Premier Chou En-lai")

or

(3) the one-letter Spanish word "y" (e.g., "President Jose Bustamante y Rivero")

or

(4) if it is one of the Arabic words "ibn", "el", "al", "er", and so forth (as in "Abdul-Assiz ibn-Saud", "Abd-el Kader", "Abd-al-Kadir", "Abd-er-Rahman").

The end of a name string is often marked by its last element (e.g., "Jr." as in "Rev. Dr. Martin Luther King Jr.", "2nd" as in "Douglas MacArthur 2nd", the Roman numerals "I", "II", "III", etc., as in "King Idris I"). Cases in which Roman numerals are in the middle of a name are rare and can be treated as listable exceptions (e.g., "King Gustaf VI Adolf").

NOTE: The present rule makes no provisions for distinguishing Roman numerals "I", "V", and "X" from the first person pronoun "I" and from the letters "V" and "X" since contexts in which these ambiguities may cause error in name recognition seem rare (e.g., "Malcolm X", "Pope Leo X", "Idris I", "May I leave?").

Ordinarily, a left parenthesis or a left bracket are among the punctuations which mark the end of a name string. This, however, is not the case when a person's title and given name are followed by his nickname in quotation marks, as for example in "Gen. Howell ("Howling Mad") Smith", "Adm. William ("Bull") Halsey", etc.

Similarly, whereas ordinarily a left bracket marks the end of a name string, occasionally, when quoting someone, newspapers supply in brackets the part of name which the original statement omitted (e.g., "My agreement with Senator [Richard B.] Russell..."); sequences such as these do not mark the end of a name.

The rule for identifying nicknames which was selected as a reasonable first approximation states that "strings of words in parentheses and quotes which occur immediately after the title and/or given names and before a surname are nicknames".

A parallel rule serves to identify names in brackets which act as amplifications of original statements.

The preceding section has stated in considerable although by no means full detail some rules for identifying titles and names. We hope that this form of presentation indicates the vast amount of detail involved in rules for automatic recognition without, however, overburdening the reader with a multitude of minute points of information.

Results of the Experiment.

Since our identification rules were embodied in dictionary entries and flow charts which were sufficiently detailed to permit an accurate manual execution of identification procedures, it was decided that our identification system would be tested out by hand on a sample of The New York Times texts.

Identification procedures were applied manually to some 40,000 words of texts. Altogether eighty-eight articles from eleven issues were selected and processed. Only newsarticles were included in the sample. All materials found in the special sections such as (1) entertainment, (2) food-fashions-family-furnishings, (3) social events, (4) necrology, etc. were omitted. Materials in the sample consisted of only texts of newsarticles; picture captions, advertisements, italicized lists of various sorts, charts and diagrams, etc. were excluded from the data.

Our 40,577-word sample contained 806 occurrences of names of persons. Of the 806 occurrences of names of persons, 46 or about 6% of the total were missed. In addition, 47 words and word strings were mistakenly identified as personal names or personal titles.

Figure of merit F for the results of this identification system was computed by means of the following formula:

$$F = \frac{C^2}{(C + M) \times T}$$

where C is the number of correct identifications, M is the number of mistaken identifications, and T is the number of names of persons in the sample. (2)

For $T = 806$, $C = 746$, and $M = 47$

$$F = \frac{746^2}{(746 + 47) \times 806} = .87$$

Analysis of Major Errors.

Twenty-six misses (out of a total of forty-six) and thirty mistaken identifications (out of a total of forty-seven) occurred in attempted identifications of words, word stems, and word strings which perform a naming function vis-a-vis some namesake (e.g., "Grumman Aircraft Engineering Corporation"). This source of misses and false identifications would be eliminated if in the future the automatic identification system was not required to decide whether words, word stems, and word strings (e.g., "Grumman") performing a naming function vis-a-vis some identifiable namesake (e.g., "Aircraft Engineering Corporation") are names of persons.

We also need more effective rules for computing namesake phrases (e.g., "Aircraft Company") and personal titles (e.g., "Fireman Apprentice") from their respective elements (e.g., "Aircraft", "Company", "Fireman", "Apprentice").

In addition, we need to prevent or eliminate the errors caused by the assumption that all capitalized words occurring after ambiguous words such as "General", "Justice", "Major", "Principal", etc. are names of persons.

We also require more effective rules to distinguish strings of titles (e.g., "President, Secretary of State") from titles followed by names. In addition, we need more effective rules for distributing a title among all names of persons which follow it in the text (e.g., "Senators Vance Hartke and Birch Bayh of Indiana and Eugene J. McCarthy and Walter F. Mondale of Minnesota").

In addition, we may require rules which would check on the old ones rather than supersede them. The new set of rules would be applied to words and phrases which were identified as names of persons by the old set of rules. The new rules could indicate the degree of confidence with which the algorithm identified a word or a string of words as name of a person, or as a personal title followed by the name of a person, etc.

The advantage of this procedure consists in not having to revamp the algorithm in order to accomodate new rules. New rules would simply be tacked on to the old ones. New rules might check whether the elements of a string of letters, punctuation marks, spaces, numbers, etc. which the old rules had identified as a name can be (a) words of the English language and (b) names of persons.

Whether a string of characters is a personal name could be decided by probability tables constructed along these lines:

Is this string of letters an English word?	Can this string of letters be a personal name?	Is this string of letters the name of a person?
Yes	Yes	Probably yes
Yes	Unknown	It's unlikely
No	Yes	Yes
No	Unknown	It's very likely

The new rules should be relatively easy to implement. The question "Is this string of letters an English word?" could be answered by means of (a) a lookup in a dictionary based on some desk dictionary -- say Webster's Collegiate, and (b) simple rules for identifying affixes of the plural, the past tense, the gerund, the negation, etc. The question "Can this string of letters be a personal name?" could be answered by means of (a) a lookup in a dictionary based on a large telephone directory -- say the Manhattan Telephone Directory, and (b) simple rules for identifying the plural (e.g., "es" and "s" as in "the Joneses" and "the Weinbergs") and other affixes.

Improving the automatic identification system may require several subsidiary investigations. For instance, we may be well advised to determine the relationship -- if any -- between, on the one hand, the length, the date, the place of origin, the subject matter, the authorship, and the type of newspaper articles processed through the system, and on the other, the effectiveness of the algorithm.

Discussion and Interpretation.

Automatic classification of words and phrases of the type described here can be regarded as a particularly simple case of machine translation. However, the goal of this type of machine translation is not translation into another natural language but TEXT REDUCTION: certain words and word strings are identified as "pertinent" (e.g., personal titles, personal names, place names, street addresses, numbers and measures, dates and other time phrases, company names, trade names, chemical formulas, etc., etc.) and others as "not pertinent". Pertinent words and phrases are retained and labeled, and all others are suppressed.

Even this simple goal requires rules which are rather complex. However, because many word strings which the algorithms such as this one attempt to recognize have simple structure ("phrase structure"), they can be generated and possibly also recognized with a reasonable degree of accuracy by simple automata ("push-down storage") or by a combination of linguistic and statistical techniques.

More generally, it may be useful to view natural language as a macro-language containing certain special-purpose micro-languages (or "sublanguages") -- each with its own structure which relative to the total structure of language is quite simple. It may be of some practical and theoretical interest (a) to investigate the structures and the inter-relations of such sublanguages and (b) to construct algorithms for identifying in texts words and word strings belonging to such sublanguages.

An ability to produce and identify automatically words and word strings belonging to various special-purpose categories (i.e., sublanguages, each with its own set of rules) may prove to be very useful in information retrieval because they play an important role in various systems for extracting and distributing information.

It would appear that along with researching and developing methods for high-quality fully automatic classification of words in texts, it may be advisable to set up efficient procedures for (a) manual classification and tagging of words and word strings in texts, and (b) subsequent automatic extraction of data from texts which were recognized either manually or automatically. One procedure for manual classification of words in texts would require computer-legible texts which can be projected on TV-type tubes (hereafter, "display screens") and either lightpens or cursors for writing on display screens. It may look approximately as follows:

A newspaper article would be copied from some type of machine-readable tape into a suitable computer. The computer would then project the article on a display screen. A clerk would then scan the display screen and locate various types of words and phrases in the article (say, names of persons, names of organizations, dates, addresses, and so forth).

Upon identifying a type of word or of word string, the clerk would flash a lightpen or a cursor at the display screen and bracket that word or word string in suitable identifying symbols. Next, identifying symbols would be transferred from display screen to tape by means of a computer program. The recognized tape could then be processed in various ways by miscellaneous information extracting programs.

It seems likely that manual assignment of word strings in texts to special-purpose sublanguages (akin to thesaurus classes) would provide a valuable interim service while methods for high-quality automatic classification are researched and developed. If and when automatic procedures for recognizing in texts dates, personal titles, various technical and professional terms, metalinguistic terms, names, etc., etc. become competitive with manual ones, the data processing community will be already in possession of operational computer programs capable of extracting data from recognized texts.

Some Possible Applications.

In the absence of figures on the cost of identifying personal titles and names by computer, the subject of the applications of computer programs capable of recognizing names of persons in newspaper texts must remain in the domain of speculations.

We would conjecture that if the speed of computation was high and its price could be kept low, and if the figure of merit could be raised to .98 or higher, then a computer program for identifying names of persons in texts would be worth incorporating into existing information retrieval systems of very large newspapers and periodicals.

It is still unknown whether a program with a figure of merit lower than .98 would be useful in information retrieval. We would surmise that it might be adequate for some purposes provided that it is sufficiently fast and cheap.

Several uses suggest themselves immediately for computer programs capable of identifying cheaply, rapidly, accurately, and exhaustively the names and titles of persons in computer-legible newspaper texts. They seem to fall into five broad and overlapping categories: (1) automatic indexing of newspaper articles, (2) determining how the names of persons cluster with one another and with other words, (3) establishing frequency counts of names of persons, (4) tracing associations between names of persons, and (5) answering questions of the "Who?" type within an automatic or semi-automatic system capable of providing answers to "Who?" "Whom?" "Whose?" "When?" and "Where?" types of questions addressed to a newspaper file.

Systems for (a) either automatic or manual classification of words and word strings in texts, and (b) subsequent automatic extraction of data from texts which were recognized either automatically or manually may be useful to many groups, among them: (1) political scientists, sociologists, lexicographers, onomasticians, and literary scholars concerned with the occurrence of names, titles, and other words in texts, (2) editors, documentalists, librairians, and others concerned with automation of editing and of literature searching, (3) opinion survey and market research statisticians concerned with the occurrence of names in texts, celebrity ratings, measurement of opinion trends, etc.

Footnotes.

(1) Borkowski, C.G., A System for Automatic Recognition of Personal Names in Newspaper Texts, Report RC-1563, Watson IBM Research Center, Yorktown Heights, N.Y., 1966, 62 pp.

(2) Meetham, A.R., Preliminary Studies for Machine Generated Vocabularies, Language and Speech, 6 (Part 1): 22-36 (January-March 1963).

Centre National de la Recherche Scientifique
SECTION D'AUTOMATIQUE DOCUMENTAIRE

Com./29/1967/ J.V. A.B./Grenoble

PROBLEMES SYNTAXIQUES DE L'INDEXATION AUTOMATIQUE DE DOCUMENTS

Andrée Borillo
Jacques Virbel

- Mai 1967 -

PLAN

0 - Introduction

1 - Définition de la représentation documentaire visée

- 1.1 - L'indexation automatique
- 1.2 - La représentation syntaxique

2 - Principes de la méthode utilisée pour l'analyse syntaxique

3 - L'analyse des textes en langage naturel

- 3.1 - Dimension de l'unité de texte analysé
- 3.2 - Mécanisme des règles
  - 3.21 - Reconnaissance de la catégorie grammaticale
  - 3.22 - Attribution du rôle syntaxique

4 - Règles de construction des syntagmes de type consécutif et comparatif

- 4.1 - Description des mots-outils
- 4.2 - Les schémas syntaxiques
- 4.3 - Exemples de classes d'outils
- 4.4 - Repérage des correlanda

5 - Construction des syntagmes de type associatif

6 - Utilisation des informations sémantiques dans les règles de construction des syntagmes

- 6.1 - Pour la construction des syntagmes consécutifs et comparatifs
- 6.2 - Pour la construction des syntagmes associatifs

7 - Résultats obtenus et leur synthèse

- 7.1 - Types de résultats
- 7.2 - Opérations de raccordements

8 - Conclusion

0 - Introduction .

On se propose d'aborder dans cette communication l'aspect syntaxique de l'indexation automatique de résumés scientifiques (1). On définit tout d'abord un modèle syntaxique propre à représenter les relations qui unissent les descripteurs extraits d'un texte en langage naturel (§ 1). Puis on expose la méthode générale que l'on compte mettre en oeuvre pour opérer le passage de l'expression en langage naturel vers cette représentation artificielle (§ 2). On définit ensuite les divers outils linguistiques utilisés à cette fin : une analyse syntaxique (§ 3) ; un ensemble de règles permettant la mise en corrélation des descripteurs ; le "réseau notionnel", organisation syntagmatique des descripteurs du langage documentaire (§§ 4-5). On montre comment ces outils utilisent des informations tout à tour syntaxiques et sémantiques (§ 6). Des opérations de synthèse sont enfin prévues pour ordonner et compléter les résultats terminaux (§ 7).

---

(1) Ces résumés, rédigés en français, ont été extraits du "Bulletin signalétique du C.N.R.S." et se rapportent au domaine de la Psycho-Physiologie.

1 - Définition de la représentation documentaire visée

Avant de décrire la représentation syntaxique que l'on vise, donnons une définition de l'indexation automatique, telle qu'elle est ici conçue.

1.1 - L'indexation automatique

L'indexation automatique est envisagée ici comme un cas particulier de traduction de textes écrits dans une langue naturelle vers une représentation de ces textes formée dans un langage documentaire artificiel. Ce langage documentaire, défini a priori, est destiné à exprimer le contenu des documents sous une forme de nature à faciliter les opérations de recherche des informations.

Les caractéristiques du langage documentaire ont déjà été décrites par ailleurs (1) ; rappelons simplement qu'il est constitué par un lexique, ensemble de "descripteurs" organisés sémantiquement - chaque descripteur correspondant à un concept défini dans un champ scientifique donné - et par une syntaxe qui fait l'objet du paragraphe suivant.

La phase de traduction syntaxique dont il sera question est précédée d'une phase dite lexicale, où l'on cherche à repérer les concepts attestés dans le texte naturel de départ et à les traduire par les descripteurs correspondants, ceci à l'aide d'un dictionnaire automatique. Nous supposerons achevée cette phase lexicale de l'indexation (2) ; la phase syntaxique qui suit s'applique donc à une liste inorganisée de descripteurs obtenus à partir du texte.

1.2 - La représentation syntaxique

Les relations syntaxiques choisies pour exprimer les rapports logiques qui unissent les descripteurs sont empruntées à la grammaire SYNTOL (3). Elles sont ici au nombre de trois :

- la relation "consécutive" qui marque une interdépendance d'ordre dynamique entre deux concepts.

Ex. : A exerce une action sur B

---

(1) Voir bibliographie, in fine, n° 1
(2) Sur cette étape, voir bibliographie n° 2
(3) Voir bibliographie n° 1, pp 44 - 88

- la relation "comparative" qui indique un rapprochement formel entre deux termes.

Ex. : A est différent de B

- la relation "associative", marquant un rapport d'ordre statique, quelle que soit sa nature : appartenance, inclusion, qualification, etc.

Ex. : A est dans B
A est accompagné de B
A appartient à B etc...

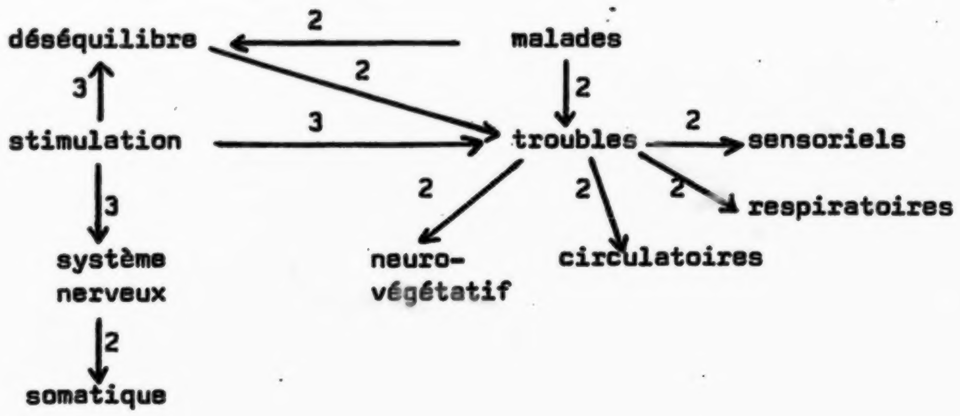
On appelle syntagme un couple de descripteurs unis par une relation. Chaque relation est orientée d'un pôle à l'autre du syntagme selon ces règles précises (1).

Si les textes analysés sont homogènes, c'est-à-dire s'ils se réfèrent à un seul thème, comme c'est généralement le cas pour des résumés d'ouvrages de psycho-physiologie, on peut poser pour principe que la représentation documentaire de ce texte doit avoir la forme d'un graphe connexe, d'où tout isolat est donc exclu.

Voici un exemple de texte que l'on se propose d'analyser, et la représentation documentaire que l'on vise à obtenir.

Texte : "Etude de trois malades présentant un déséquilibre unilatéral avec troubles sensoriels après diverses stimulations. Ces dernières provoquent aussi des troubles neuro-végétatifs, circulatoires et respiratoires, montrant que le système nerveux somatique n'est pas seul intéressé par les stimuli".

Représentation documentaire visée


déséquilibre
2
malades
2
3
2
stimulation
3
troubles
2
sensoriels
3
2
2
2
respiratoires
système
nerveux
neuro-
végétatif
circulatoires
2
somatique

---

(1) Voir bibliographie nº 1, pp 51 - 66

Les termes soulignés dans le texte original sont ceux qui ont un équivalent lexical dans le langage documentaire. Ce sont ces descripteurs que l'on trouve dans la représentation documentaire (1). Les relations syntaxiques attestées entre les descripteurs sont notées par une flèche ($\xrightarrow{3}$: relation consécutive ; $\xrightarrow{2}$: relation associative).

Comme on le voit, la représentation obtenue ne compte pas d'isolat, mais constitue au contraire un graphe unique.

C'est ce type de produit que l'on s'efforce d'obtenir par la procédure décrite aux paragraphes suivants.

2 - Principes de la méthode utilisée pour l'analyse syntaxique

L'analyse syntaxique consiste donc à passer d'un texte en langage naturel vers la représentation documentaire telle qu'elle vient d'être définie. Un certain nombre d'hypothèses de travail ont été posées pour élaborer les règles nécessaires à ce passage.

a) Tout d'abord, on part du postulat que les relations consécutives et comparatives sont marquées dans le langage naturel, pris dans un domaine spécialisé, par des constructions dont on peut donner des listes finies. A la base de ces constructions, on constate qu'apparaissent des connecteurs syntaxiques ou mots-outils que l'on peut isoler et caractériser. A partir de ces repères que constituent les mots-outils, on estime pouvoir établir des règles de corrélation pour la production de syntagmes corrects et non ambigus. Par ce procédé toutes les relations consécutives et comparatives établies entre descripteurs du langage documentaire peuvent être extraites d'un texte et fournir une partie de la représentation documentaire visée.

b) De ce fait, on déduit que toutes les relations qui restent à extraire sont des relations associatives. Cette démarche qui fait de la relation associative une relation résiduelle peut se justifier dans la mesure où cette relation étant la plus chargée d'un point de vue sémantique et syntaxique, il est préférable de ne pas avoir à l'interpréter mais de l'obtenir par déduction.

Cependant même si l'interprétation de la relation associative est implicite, il faut au moins la détecter dans le texte, et il semble qu'on ne puisse opérer comme précédemment : reconnaître tout d'abord un mot-outil susceptible d'expliciter la relation, puis d'après l'outil, rechercher les correlanda. Etant donné la variété de signification que la relation associative recouvre, les types d'outils à recenser seraient très nombreux et difficiles à caractériser. D'autre part, même si l'on réussissait à dresser une liste quasi exhaustive de ces mots-outils et à les classer selon certains critères opératoires, il n'est pas certain que le repérage des correlanda en serait facilité. En effet, les mots-outils considérés étant de type très différent, les contextes syntaxiques dans lesquels ils apparaissent n'ont rien de commun - à la limite on aboutit presque à des cas d'espèces qu'il faut relever un à un. De plus le contexte syntaxique n'étant pas suffisant, dans beaucoup de cas il faut avoir recours au contexte sémantique des correlanda eux-mêmes, qui de ce fait jouent un rôle aussi important que les mots-outils recensés. C'est pourquoi pour la construction des syntagmes de type associatif une démarche différente est prévue. On n'utilise pas de mots-outils comme précédemment ; on relève les descripteurs qui n'ont pas été mis en relation consécutive ou comparative et on établit entre eux des corrélations de fonction à fonction du type : "nom sujet - verbe" ou encore "nom sujet - nom complément d'objet".

---

(1) Malgré l'identité orthographique que l'on n'a maintenue que pour plus de clarté.

c) Une autre hypothèse a été posée concernant la place et l'importance relatives des informations de type sémantique et syntaxique dans la méthode d'analyse. Lors d'une expérience antérieure, une première méthode d'analyse avait été expérimentée (1); elle se caractérisait en particulier par l'usage exclusif d'outils sémantiques pour la mise en corrélation : le lexique et le réseau notionnel.

Le réseau notionnel constitue une organisation syntagmatique des descripteurs du lexique, et atteste ainsi les affinités combinatoires des concepts ; par exemple, la classe des agents (chimiques, physiques, psychologiques, etc.) est reliée par une relation consécutive aux entités pouvant être affectées par leur action (cerveau, foie, digestion, sécrétion, etc...). Dans cette méthode, la construction des syntagmes se déclenche au vu de mots-outils, termes ayant un rôle syntaxique dans la langue naturelle. Elle s'opère par consultation du réseau notionnel qui atteste quels couplages des descripteurs sont acceptables dans le contexte du mot-outil.

Les résultats ont montré (2) d'une part que cette méthode permet de résoudre nombre de difficultés lors de l'établissement des relations entre descripteurs, et d'autre part que les erreurs constatées peuvent souvent être réduites par l'apport d'informations syntaxiques.

En utilisant comme base cette première méthode, on se propose donc de ne pas limiter les moyens de l'analyse à des informations strictement sémantiques, et d'exploiter des informations grammaticales et fonctionnelles du contexte en langage naturel. Ainsi, en diversifiant la nature des informations, et en les combinant, on vise à mettre au point une méthode mixte, plus efficace que chacune des méthodes syntaxique ou sémantique prise séparément.

---

(1) Voir bibliographie nos 3 et 4.
(2) Voir l'évaluation des résultats dans bibliographie nº 4.

d) Cette exploitation des informations autres que sémantiques suppose une procédure d'analyse grammaticale du texte en langage naturel. En effet pour pouvoir utiliser le contexte syntaxique, il faut qu'un traitement autonome ait fait attribuer à chacun des mots de l'énoncé où se trouve l'outil syntaxique ou les descripteurs, le type de renseignement grammatical et fonctionnel que l'on attend pour procéder à la recherche des correlanda.

Une dernière hypothèse, d'ailleurs liée à la précédente, est posée en ce qui concerne la nature et les dimensions de cette analyse. L'exploitation que l'on veut faire des résultats de cette analyse grammaticale, et le rôle que l'on entend faire jouer aux informations sémantiques grâce au réseau notionnel, permettent d'avancer qu'il n'est pas nécessaire d'effectuer une analyse totale de la phrase, mais qu'une simple segmentation en syntagmes caractérisables du point de vue fonctionnel est suffisante.

C'est cette première opération d'analyse qui est exposée au paragraphe suivant (§ 3). A l'issue de cette analyse, les renseignements grammaticaux et fonctionnels étant obtenus, on aborde la phase de construction des syntagmes. Etant donné les procédures distinctes préconisées pour l'obtention des relations consécutives et comparatives d'une part, pour les relations associatives d'autre part, on examine tout d'abord la construction des syntagmes du premier type (§ 4), puis celle des syntagmes du second type (§ 5), en indiquant (§ 6) quelle est la part des informations sémantiques dans chacune des constructions. Dans un dernier paragraphe (§7), on donne un aperçu général des résultats obtenus par ces divers procédés et des solutions envisagées pour parfaire la représentation documentaire: complètements et raccordements.

3 - L'analyse des textes en langage naturel

3.1 - Dimension de l'unité de texte analysé

Une segmentation des textes est déjà faite dans la phase de traduction lexicale, nécessitée par le balayage d'unités de contexte déterminées lors de l'interprétation sémantique des termes polysémiques (1). Ces unités contextuelles sont délimitées par le point -lorsqu'il n'est pas un signe d'abréviation - et le point-virgule. Pour la traduction syntaxique, ce premier découpage fait apparaître des unités trop complexes, car elles contiennent des propositions de nature différente (principales, subordonnées, complétives, etc.) ; on donne donc également le statut de séparateur aux conjonctions de subordination. L'analyse se déclenche et s'arrête alors au vu de l'un des séparateurs suivants : signe de ponctuation ou conjonction de subordination.

---

(1) Voir bibliographie n° 2

Cependant si les opérations d'analyse se déroulent de manière discontinue et traitent chaque membre de phrase comme une unité autonome, il ne faut pas oublier que le résultat final attendu doit perdre ce caractère disjoint pour former une représentation cohérente du texte. Nous verrons (§ 7.2) les opérations qui permettent de relier entre eux les groupes de syntagmes isolés en raison du caractère fragmentaire de l'analyse.

3.2 - Mécanisme des règles

L'analyse, s'appliquant à des membres de phrase compris à l'intérieur des frontières que nous venons de définir, s'effectue grâce à des règles de concaténation. Ces règles fixent l'enchaînement des catégories grammaticales des mots pris deux à deux dans l'ordre où ils apparaissent dans la phrase examinée de gauche à droite. Les règles établies sous forme d'équivalences comportent deux parties : la partie droite concerne le choix de la catégorie grammaticale, la partie gauche l'attribution du rôle syntaxique.

3.21 - Reconnaissance de la catégorie grammaticale

La possibilité de juxtaposition de deux catégories dans une phrase obéit à des règles de distribution. En faisant le recensement de toutes les juxtapositions permises, on arrive à résoudre certaines ambiguités dues à des homographies.

Ex. : Le fait de trouver une règle : "article + nom = ...", mais non pas : "pronom + nom = ...", résoud l'ambiguité du type : "Le chat" ... "Le", "pronom" ou"article", placé immédiatement avant un nom ne peut être qu'"article".

Le filtrage n'est pas toujours aussi facile :

Ex. :	Le	chat	entre	dans	la	cage
	"pro.pers."	"nom"	"verbe"	"préposition"	"pro.pers."	"nom"
	"article"		"préposition"		"article"	

D'après la liste de règles établie, on peut éliminer l'interprétation "pronom personnel" pour "le", mais pour "chat" et "entre" correspondent deux règles de caractérisation :

nom sujet + verbe = ...
nom sujet + préposition = ...

(le premier constituant est accompagné d'un code fonctionnel attribué lors de la règle précédente, le deuxième ne comprend encore que le code de catégorie).

Etant donné les deux règles permises, l'interprétation de "entre" peut être aussi bien "verbe" que"préposition". Nul choix n'étant possible, les deux interprétations sont donc conservées.

A l'étape suivante, lorsque l'on doit faire l'enchaînement "entre"+"dans", une seule règle est possible :

verbe intransitif + préposition =

La double interprétation pour "entre" n'est plus possible, celle de "préposition" est rejetée puisqu'elle n'est plus vérifiée par ce qui suit. Après examen de toute la phrase, on obtient en ce qui concerne les catégories :

le	chat	entre	dans	la	cage
art.	nom	verbe	prép.	art.	nom

3.22 - Attribution du rôle syntaxique

Dans la deuxième partie de la règle est définie la fonction syntaxique du deuxième constituant.

Ex. : verbe transitif * + nom = verbe transitif + nom compl. objet (l'astérisque qui accompagne "verbe transitif" dans la première partie de la règle indique que le verbe n'a pas encore reçu de complément d'objet ; ce signe disparaît dans la deuxième partie puisque le complément d'objet est trouvé).

Le rôle syntaxique du nom découle ici de la nature et du rôle du mot qui précède le verbe transitif - qui lui-même a été défini par rapport au mot qui précédait, etc...

Ex. : (1) le médicament provoque des troubles
Ex. : (2) les troubles que provoque le médicament

Bien que placés de même façon par rapport à "provoque", "troubles" et "médicament" n'ont pas la même fonction. Cette différence est marquée explicitement par le verbe :

Dans (1) verbe transitif * + nom = verb. transit. + c. objet
Dans (2) *verbe transitif + nom = verb. transit. + sujet

Dans (2) la présence de l'astérisque avant le verbe indique que le sujet n'a pas encore été trouvé ; l'absence d'astérisque, après, signale par contre que le complément d'objet précédait le verbe ; "troubles" a donc une fonction de "complément d'objet", "médicament" une fonction de "sujet".

Parfois la catégorie et la fonction d'un mot ne suffisent pas à déterminer la fonction du mot qui suit. C'est le cas par exemple des compléments prépositionnels :

(3) Etude sur les souris de l'antagonisme...
(4) Etude sur le foie de la souris...

Ici aucune règle ne peut montrer que "de l'antagonisme" dépend de "étude" alors que "de la souris" dépend de "foie". Au lieu de formuler une règle arbitraire qui ferait par exemple dépendre un complément prépositionnel du premier mot qui le précède, on a choisi de faire du complément prépositionnel, lorsqu'il ne suit pas un nom sujet, une structure autonome dont la nature est indiquée par un code spécifique.

Ex. : Etude sur les souris / de l'antagonisme /: "compl. prépositionnel" (1)
Ex. : Etude sur le foie / de la souris /: "compl. prépositionnel"

De la même façon, des indices spécifiques ont été choisis pour les membres de phrase coordonnés.

Ex. : Etude de l'activité bioélectrique du cerveau et des troubles du comportement.

On n'est pas autorisé à établir systématiquement une liaison entre les éléments les plus rapprochés de part et d'autre de la coordination. Ici les deux noms "cerveau" et "troubles", même introduits par les mêmes prépositions "de", ne sont pas mis en liaison par "et". Cette liaison est en fait entre "activité" et "troubles".

C'est pourquoi on préfère en général ne pas essayer de rattacher le membre de phrase coordonné. On le considère comme disjoint et on lui donne une spécification qui le fait reconnaître des autres structures autonomes (les compléments prépositionnels par exemple). Le seul cas particulier est celui des adjectifs qui, lorsqu'ils sont de part et d'autre d'une conjonction de coordination, sont reliés et reçoivent le même code de fonction.

Ex. : des lobes frontaux et latéraux

Mais : Etude de l'activité bioélectrique / du cerveau /
/ et des troubles / du comportement
"membre de phrase coordonnée"

On agit de même pour les prépositions relatives, certains membres de phrase isolés par des virgules, etc.

- La dépression du rythme alpha / qui est
"groupe nominal sujet" prop.

caractéristique / de la réaction / d'orientation /
"relative" "compl.préposit." "compl.préposit."

reflète un processus / d'excitation /
"groupe verbal" "compl. préposit."

---

(1) Dans les exemples qui suivent, le "/" indique les points de segmentation de la phrase.

Comme on vient de le montrer, si l'on considère l'arborescence que l'analyse structurale de la phrase devrait engendrer, on s'arrête à des résultats intermédiaires, loin du sommet.

On va voir comment les informations fournies par l'analyse grammaticale sont reprises et utilisées dans les constructions des syntagmes SYNTOL.

4 - Règles de construction des syntagmes de type consécutif et comparatif

La construction s'effectue à partir des mots-outils. Nous verrons en premier lieu ce que sont les mots-outils, et comment ils sont spécifiés (§ 4.11-4.13) ensuite comment, suivant les règles attachées aux mots-outils, se déroule la recherche des descripteurs à corréler (§ 4.14).

4.1 - Description des mots-outils

Les mots-outils sont des mots ou groupes de mots du langage naturel qui marquent des liaisons de type consécutif ou comparatif.

On peut les distinguer du point de vue :

a) grammatical : ils peuvent être des verbes, des noms, des adjectifs, des prépositions.... Voci quelques mots-outils de type consécutif :

verbes : inhiber, provoquer, augmenter...
noms : suppression, effet de, réaction à....
adjectifs : épileptogène, destructeur, consécutif à....
prépositions : par suite de, en fonction de...

b) formel : certains comportent un seul terme (ex. : provoquer, épileptogène...) d'autres peuvent être fragmentés et admettre des insertions entre leurs différents constituants (réagit le plus souvent à..., a un effet passager sur...); d'autres enfin se présentent comme nécessairement discontinus (action de l'atropine sur le foie, réaction du muscle à la stimulation...). Dans une première expérience (1), une distinction très importante du point de vue opératoire avait été faite entre les premiers, les outils autonomes, et les derniers, les grilles, cas particulier d'outils discontinus dont les insertions pouvaient être des descripteurs. Cette distinction n'est pas retenue dans la nouvelle procédure.

---

(1) Voir bibliographie nº 3

c) sémantique : certains mots-outils reçoivent une traduction en SYNTOL: ils sont à la fois descripteurs et connecteurs, on dit qu'ils sont mixtes. D'autres ne sont pas retenus pour leur signification mais seulement pour leur fonction dans la phrase, ce sont les outils purs.

Deux critères sont tout d'abord retenus pour la classification des outils : la catégorie grammaticale et l'interprétation lexicale. D'après ces deux critères, on peut dresser une liste des mots-outils de type consécutif.

	Catégorie grammaticale				Interprétation lexicale	
	verbe	nom	adjectif	préposit.	mixte	pur
inhiber	X				X	
provoquer	X					X
modification		X			X	
effet		X				X
epileptogène			X		X	
par suite de				X		X

Dans cette classification, on remarque que seuls les verbes et les noms sont soit mixtes, soit purs, tandis qu'on n'a pas trouvé de mots-outils adjectifs qui ne soient pas mixtes, d'outils prépositions qui ne soient pas purs.

Cependant la classification des mots-outils ne se fonde pas seulement sur la combinaison de ces deux critères. Une distinctions supplémentaire est retenue qui fait appel au type de construction syntaxique du mot-outil, c'est-à-dire à la nature grammaticale et la fonction syntaxique des termes qu'il est susceptible de mettre en relation. Ce troisième critère de spécification ne va pas donner lieu à une nouvelle dichotomie des classes déjà indiquées, mais va conduire à reconsidérer les distinctions faites suivant les catégories grammaticales. En effet, la nature des compléments d'un mot dépend en premier lieu de sa catégorie. Ex. : un nom peut être complété par un adjectif ou un autre nom, un verbe attend un sujet généralement nom ou pronom, des compléments directs ou indirects introduits par des prépositions etc... Ensuite, à l'intérieur de chacune des catégories grammaticales des différences se font jour : Ex. les verbes transitifs sont complétés par des noms sans l'intermédiaire de prépositions, ils peuvent se construire à la forme passive, certains verbes intransitifs régissent certaines prépositions etc...

C'est cette notion de différenciation syntaxique que l'on va exprimer sous forme de règles qui, attachées à une sous-classe de mots-outils, indiqueront toutes les combinaisons syntaxiques possibles pour ce type d'outil. A ces combinaisons, on a donné le nom de "schémas syntaxiques".

4.2 - Les schémas syntaxiques

Les schémas syntaxiques sont des suites continues ou discontinues dans le langage naturel de codes grammaticaux et fonctionnels énoncés dans un ordre donné avant et/ou après l'outil, dans les limites d'une zone déterminée, sans spécification de distance par rapport à l'outil.

Ex. : Au mot-outil de type consécutif : "effet" (de... sur...) est associé une série de schémas syntaxiques dont voici quelques exemples (1).

$$\underline{\text{Nom}} \quad \left\{\begin{array}{c}\text{sujet}\\ \text{ou}\\ \text{compl.}\end{array}\right\} \quad + \text{ de } + \text{ compl. prépositionnel } + \text{ sur } + \text{ compl. prépositionnel}$$

$$\underline{\text{Nom}} \quad \left\{\begin{array}{c}\text{sujet}\\ \text{ou}\\ \text{compl.}\end{array}\right\} \quad + \text{ sur } + \text{ compl. prépositionnel } + \text{ de } + \text{ compl. prépositionnel}$$

$$\text{Nom} \quad \text{sujet} \quad + \text{ avoir } + \underline{\text{nom compl. objet}} + \text{ sur } + \text{ compl. prépositionnel}$$

$$\text{Nom} \quad \left\{\begin{array}{c}\text{sujet}\\ \text{ou}\\ \text{compl.}\end{array}\right\} \quad + \text{ qui } + \text{ avoir } + \underline{\text{nom compl.objet}} + \text{ sur } + \text{ compl.prépositionnel}$$

$$\text{Nom} \quad \left\{\begin{array}{c}\text{sujet}\\ \text{ou}\\ \text{compl.}\end{array}\right\} \quad + \text{ avoir } + \underline{\text{nom compl. objet}} + \text{ adjectif}$$

---

(1) Dans ces exemples, le code grammatical et fonctionnel représentant le mot-outil est souligné.

Ex. : sujet : L'effet de l'atropine sur le foie est étudié...

compl. d'objet : On étudie l'effet de l'atropine sur le foie....

compl. prépositionnel : Etude de l'effet de l'atropine sur le foie...

Pour un autre mot-outil, "provoquer" par ex., la liste des schémas est tout à fait différente :

Nom Sujet + Verbe + compl. d'objet	l'hexorbarbital provoque le sommeil
Nom Sujet + qui + Verbe + compl.objet	l'hexorbarbital qui provoque le sommeil
Nom Sujet ou compl. + que + Verbe + sujet	le sommeil que provoque l'hexorbarbital
Nom Sujet + être + Part. passé + par + compl. préposition	le sommeil est provoqué par l'hexorbarbital

Pour pouvoir caractériser de la sorte un mot-outil, il faut qu'il apparaisse dans des contextes syntaxiques suffisamment stéréotypés. C'est le cas en général des verbes de types consécutifs ou comparatifs qui sont soit transitifs directs et impliquent un complément d'objet, soit des verbes transitifs indirects tels que "agir sur", "dépendre de"....

Les noms eux aussi ont une construction stable ; de même que les verbes à gouvernant fort, ils appellent certaines prépositions :

dans
effet de... sur ...
chez

Il est possible de dresser une liste finie des prépositions ou locutions prépositives pouvant servir de charnière à un mot-outil donné ; ici "effet" peut se construire avec l'une des prépositions indiquées "dans, sur, chez".

Dans les schémas syntaxiques donnés en exemple plus haut, la stricte juxtaposition des éléments requis n'est pas obligatoire. Un nom Sujet peut être séparé du verbe par des compléments de nom, des adjectifs, des adverbes.... mais il est reconnaissable ; sa qualité de sujet le rend caractérisable quelle que soit la distance dans la phrase qui le sépare du verbe. De même le complément d'objet peut être séparé du verbe.

Ex. : "L'hexorbarbital, administré à faible dose, provoque généralement le sommeil".

Certains mots-outils sont des prépositions ou des locutions prépositives : "par suite de", "à cause de", "à la différence de". Pour ce type d'outil il est difficile d'établir des schémas syntaxiques rendant compte des termes mis en relation. On peut connaître le correlandum qui suit la préposition, mais l'autre peut avoir un emplacement très variable.

"Troubles de mémoire par suite de traumatisme"
"Par suite de traumatisme le malade a perdu la mémoire"
"Le malade devient amnésique par suite de traumatisme"

C'est pourquoi, dans les cas où on ne peut trancher, on n'établit, pour les prépositions qu'un schéma syntaxique incomplet du type :

	Ex. :
préposition + nom	"à la suite de contusions"
préposition + infinitif	" à force de parler"

4.3 - Exemple de classes d'outils

Les schémas syntaxiques ont été utilisés pour affecter les outils à des classes distinctes. Cependant comme critère d'affection on n'a pas pris le schéma syntaxique mais sa condensation : le schéme syntaxique. Au lieu de contenir des codes spécifiés comme le schéma, le schéme comprend des macro-codes qui représentent une généralisation au niveau de la catégorie grammaticale des mots :

"Nom + préposition + nom + préposition + nom" est le schéma correspondant à la fois à :

$$\text{Nom} \begin{Bmatrix} \text{compl.} \\ \text{ou} \\ \text{sujet} \end{Bmatrix} + \text{de} + \text{nom} + \text{sur} + \text{nom}$$

$$\text{Nom} \begin{Bmatrix} \text{compl.} \\ \text{ou} \\ \text{sujet} \end{Bmatrix} + \text{à} + \text{nom} + \text{par} + \text{nom}$$

Ainsi on peut dire que "effet de... sur...", "réaction à... par" ont le même schéme syntaxique.

Voici le tableau de quelques mots-outils de type consécutif rangés suivant leur schéme syntaxique.

	catégorie de l'outil				cat. et/ ou fonction des correlenda					nature lexicale de l'outil	
	verbe	nom	adj.	préposit.	nom avant	nom après	adj. après	compl. préposit. 1	compl. préposit. 2	mixte	pur
provoquer	X				X	X					X
inhiber	X				X	X				X	
agir	X				X			X			X
employer	X					X		X			X
régulation		X						X	X	X	
réaction		X						X	X		X
cause		X			X			X			X
origine		X					X				X
					X		X	X			
action		X						X	X		X
					X		X	X			
sensible			X		X			X		X	
consécutif			X		X			X			X
épileptogène			X		X					X	
par suite de				X			X				X

Un même type de classification a été élaboré pour les mots-outils de type comparatif qui se répartissent suivant une dizaine de classes environ.

Cette classification va faciliter la recherche des descripteurs entrant dans une relation de type consécutif ou comparatif.

4.4 - Repérage des correlanda

Le repérage des correlanda s'opère grâce à des règles de corrélation qui accompagnent chacun des schémas syntaxiques attachés aux types de mots-outils. Ces règles indiquent non seulement quels sont les termes des schémas qui sont mis en relation mais aussi dans quel sens s'oriente la relation.

Exemple, pour "provoque" et "inhibe" (quelques règles seulement) :

type d'outil	Code	schéma syntaxique	règle (1)	Vérif. sémantique
provoque	VT	nom sujet + verbe + nom compl. objet	$1 \xrightarrow{3} 3$	
	VT	nom sujet + qui + verbe + nom compl.objet	$1 \xrightarrow{3} 4$	
	ST	nom $\left\{\begin{matrix}\text{compl.}\\ \text{ou}\\ \text{sujet}\end{matrix}\right\}$ + part.passé+prép.PAR+nom compl.	$1 \xleftarrow{3} 4$	S
inhibe	VT	nom sujet + verbe + nom compl.objet	$\begin{tikzcd} & 2 \arrow[dr, "2"] & \\ 1 \arrow[ur, "2"] \arrow[rr, "3"] & & 3 \end{tikzcd}$	
	VT	nom sujet+qui+verbe+ nom compl.objet	$\begin{tikzcd} & 3 \arrow[dr, "2"] & \\ 1 \arrow[ur, "2"] \arrow[rr, "3"] & & 4 \end{tikzcd}$	
	ST	nom $\left\{\begin{matrix}\text{compl.}\\ \text{ou}\\ \text{sujet}\end{matrix}\right\}$ +part.passé+prép.PAR+nomcompl.	$\begin{tikzcd} & 3 \arrow[dl, "2"'] & \\ 1 & & 4 \arrow[ul, "2"'] \arrow[ll, "3"] \end{tikzcd}$	S

(1) Le chiffre à gauche de la flèche représente le numéro d'ordre dans le schéma syntaxique du 1er terme mis en relation, le chiffre à droite de la flèche celui du 2ème terme.
Ex. : $1 \xrightarrow{3} 3$ signifie que c'est le "nom sujet" (1er terme du schéma) et le "nom complément d'objet" (3ème terme du schéma) qui sont mis en relation de type 3 (relation consécutive).

La différence entre les règles de construction de "provoque" et "inhibe" vient du carctère mixte du deuxième outil : en plus de la relation consécutive entre "nom sujet" et "nom compl.objet" s'établissent deux relations associatives entre l'outil lui-même et les deux correlanda "nom sujet" et "verbe", et "verbe" avec "compl. objet".

Ce tableau peut se lire ainsi : si dans une phrase du texte soumis à l'analyse apparaît un outil de type "provoque", suivant le code attribué à cet outil par l'analyse grammaticale (ici VT : verbe conjugué transitif ou ST participe passé), on recherche le ou les schémas syntaxiques dans lequel on peut le rencontrer. Puis, parmi les schémas possibles, on choisit celui qui contient les mêmes catégories que celles accompagnant le mot-outil dans la phrase. Suivant le type de catégories qui forment le schéma on indique la relation qui les relie en spécifiant l'orientation. Ici les règles "sujet - compl. d'objet" et "sujet - compl. d'agent" ont une orientation différente.

Pour que la construction de syntagme soit possible il faut naturellement que les deux termes mis en relation aient une équivalence en SYNTOL. Sinon la règle ne peut s'appliquer telle quelle. Deux solutions peuvent être envisagées : on peut soit accepter un syntagme incomplet si l'un des termes n'a pas reçu de traduction au cours de la phrase lexicale, soit rechercher autour du terme non traduit un qualificatif ou un complément de nom qui pourrait, lui, avoir un équivalent documentaire et faire office de correlandum à sa place, les deux solutions pouvant être d'ailleurs exploitées concurremment.

Le code "préposition" est suivi dans les exemples donnés de PAR. Ceci signifie que la mise en relation entre le premier terme du schéma et le quatrième n'est acceptée que si le troisième - la préposition - appartient à la liste PAR. Cette liste dressée à l'avance contient une série de prépositions ou locutions prépositives acceptables dans la même position d'un schéma, c'est-à-dire ici susceptible d'introduire un complément d'agent.

PAR = | par
| à l'aide de
| grâce à
| sous l'effet de
| ....

D'autres listes semblables existent, contenant des prépositions interchangeables dans des situations données.

SUR = | sur
| dans
| chez
| pour
| ....

Dans la dernière colonne "vérification sémantique", la présence d'un S - comme c'est le cas pour la troisième règle dans le tableau - signifie que les informations grammaticales fournies par le schéma syntaxique ne sont pas sûres et qu'il faut avoir recours au réseau notionnel pour vérifier la validité de la construction. En l'occurrence, la chaîne composée d'un nom introduit par une préposition, même si celle-ci appartient bien à la liste PAR, succédant à un participe passé, lui-même succédant à un nom, peut très bien ne pas représenter une chaîne du type "le sommeil provoqué par l'hexorbarbital" mais par exemple :

"(le sommeil) du malade provoqué par intervalles (par l'hexorbarbital)".

Dans les cas, où une préposition apparaît dans un schéma syntaxique, et qu'elle ne dépend pas d'un nom sujet, la vérification sémantique est appliquée pour que soit attesté le bien fondé de la relation.

Nous venons d'indiquer la démarche générale pour la construction des syntagmes de type consécutif et comparatif. Nous allons voir maintenant comment s'établit la troisième relation, la relation associative.

5 - Construction des syntagmes de type associatif

Dans notre hypothèse de travail (§ 2), nous avons indiqué que toute relation qui n'a pas été interprétée comme comparative ou consécutive doit être interprétée comme associative. Encore faut-il s'assurer qu'une relation existe, autre que consécutive ou comparative, entre les descripteurs qui n'ont pas été reliés, ou entre ceux-ci et les descripteurs entrant dans les syntagmes déjà formés.

Or, dans le cas de la relation associative on ne veut pas utiliser de mots-outils comme repères : tout d'abord parce qu'ils sont trop nombreux et hétérogènes. Ils comprennent pratiquement tous les verbes transitifs directs et indirects, toutes les prépositions, les conjonctions de coordination, les adjectifs susceptibles d'avoir un complément, etc. Il faut également ranger dans ce groupe tous les outils implicites, qui ne sont pas à proprement parler des mots-outils, mais représentent les cas où la seule juxtaposition de deux descripteurs suffit à indiquer une relation de type associatif. En général ces descripteurs correspondent à des noms et adjectifs, ex. : "alvéoles pulmonaires".

De plus, il serait long et difficile de caractériser chaque type d'outil par des schémas syntaxiques, car les constructions dans lesquelles généralement ils s'inscrivent sont mal définies.

Devant cette difficulté de caractérisation syntaxique des mots-outils, on a préféré partir des descripteurs eux-mêmes.

A l'issue des opérations de construction des syntagmes de type consécutif et associatif, un certain nombre de descripteurs du texte soumis à l'analyse n'ont pas été mis en relation. En prenant chacun de ces descripteurs un à un, on essaie de les relier à un second descripteur appartenant à la même unité contextuelle, que celui-ci soit déjà mis en relation ou pas en confrontant les codes grammaticaux et fonctionnels qui leur ont été associés lors de l'analyse syntaxique . Une liste de règle indique quels types de rapports syntaxiques sont à retenir et quelle orientation ils déterminent dans la construction des syntagmes.

Par exemple :

	règle	orientation	validité relation
verbe transitif + nom objet =	$1 \xrightarrow{2} 2$		
nom sujet + verbe =	$1 \xrightarrow{2} 2$		
nom + adjectif =	$1 \overset{2}{\text{——}} 2$	S	
nom sujet + compl. prépositionnel =	$1 \overset{2}{\text{——}} 2$	S	S
nom compl. + compl.prépositionnel =	$1 \overset{2}{\text{——}} 2$	S	S
nom compl. + nom coordonné =	$1 \xleftrightarrow{2} 2$		S

Dans ce tableau, la présence d'un S dans la colonne "Orientation" indique que les seules informations grammaticlaes et fonctionnellles ne suffisent pas à indiquer l'orientation de la relation. La règle n'indique pas l'orientation, qui ne sera établie qu'après consultation du réseau notionnel.

6 - Utilisation des informations sémantiques dans les règles de construction des syntagmes

On constate que dans la construction des syntagmes de type 2 aussi bien que pour celle des syntagmes de type 3 ou 4, les informations grammaticales et fonctionnelles fournies par le contexte naturel ne peuvent pas toujours à elles seules répondre de la validité d'une mise en relation. Les constructions jugées douteuses sont vérifiées par consultation du réseau notionnel et acceptées ou rejetées suivant le résultat de la vérification.

Il n'est pas dit que ce contrôle sémantique arrive dans tous les cas à produire des résultats corrects. Il se peut par exemple qu'une relation entre deux descripteurs soit tenue pour valide par le réseau notionnel et cependant qu'elle ne reflète pas l'idée qui veut être exprimée dans un texte. Cependant on peut considérer le contrôle sémantique comme une sorte de filtre supplémentaire ; il écarte un certain nombre d'erreurs que la seule vérification syntaxique est impuissante à éviter et dans le cas d'incertitude pèse sur le choix de l'orientation d'une relation ou de son interprétation.

Ce contrôle se fait à différentes étapes de la traduction syntaxique. Comme on l'a vu aux § 4 et § 5, on a recours au réseau notionnel lors de la construction des syntagmes lorsque le résultat semble incertain ou qu'un choix se révèle impossible du point de vue syntaxique. On l'utilise également lorsque la traduction syntaxique des unités contextuelles étant terminée, on doit faire le raccordement des différents groupes de syntagmes ou d'isolats pour former un graphe unique. Dans ce paragraphe nous n'aborderons que le premier problème, la vérification sémantique en cours d'analyse, réservant pour le paragraphe 7 l'utilisation du réseau notionnel pour les opérations de complètement.

Nous allons résumer tous les cas où la vérification sémantique se révèle nécessaire en cours d'analyse.

6.1 - Pour la construction des syntagmes consécutifs et comparatifs

a) Lorsque les critères grammaticaux sont insuffisants pour retrouver les correlanda. Cela se produit lorsque l'un des descripteurs du schéma syntaxique correspond à un complément prépositionnel (§ 4.4). Un S inscrit dans la colonne "vérification sémantique" signale la nécessité d'informations sémantiques.

Ex. : Effet sur le sommeil / de l'animal / de l'hexorbarbital / ...

Dans cette phrase il faut choisir entre

$\text{sommeil} \xleftarrow{3} \text{animal}$

et $\text{sommeil} \xleftarrow{3} \text{hexorbarbital}$

puisque les deux compléments "de l'animal" et "de l'hexorbarbital" peuvent, l'un aussi bien que l'autre, faire partie du schéma syntaxique:

$$\left| \; \underline{\text{nom}} \begin{Bmatrix}\text{sujet}\\ \text{ou}\\ \text{compl.}\end{Bmatrix} + \text{sur} + \text{nom} + \text{de} + \text{nom} \;\middle|\; 3 \xleftarrow{3} 5 \; \right|$$

En consultant le réseau notionnel on s'aperçoit que la liaison animal-sommeil ne peut donner naissance qu'à une relation associative, orientée du premier vers le deuxième correlandum :

$\text{animal} \xrightarrow{2} \text{sommeil}$

tandis que la liaison sommeil-hexorbarbital est bien de type consécutif et peut se traduire par :

$\text{hexorbarbital} \xrightarrow{3} \text{sommeil}$

Ce sont donc les deux descripteurs "sommeil" et "hexorbarbital" qui sont retenus ici dans la phrase et mis en relation.

b) La vérification sémantique est également nécessaire lorsque le mot-outil est ambigu. La relation qu'il détermine peut alors avoir plusieurs sens suivant la classe sémantique des termes mis en relation.

Ex. : emploi de la réserpine pour l'inhibition motrice
emploi d'un outillage particulier pour l'expérience...

L'outil "emploi de... pour..." détermine une relation consécutive dans un cas, alors que la relation est associative - instrumentale ici - dans la deuxième phrase.

Des outils tels que "employer pour", "après", "provenir de"... ont respectivement un sens instrumental, temporel, locatif et peuvent également avoir un sens consécutif. Pour eux, quelles que soient les règles de construction appliquées, la vérification sémantique est obligatoire, et la consultation du réseau fournit à la fois l'interprétation de la relation et son orientation.

6.2 - Pour la construction des syntagmes associatifs

C'est seulement la validité de la relation qui est à vérifier. La relation n'est pas à interpréter puisque si elle existe, on sait qu'elle est associative.

a) Comme dans le cas des autres syntagmes, la vérification sémantique se révèle nécessaire lorsque sont mis en relation des compléments prépositionnels ou des membres de phrase coordonnés.

Ex. : Réaction de croissance / à la lumière / des phycomyces.

La règle syntaxique "nom compl. + nom compl." fait construire pour cette phrase :

soit : lumière $\xrightarrow{2}$ phycomyces

soit : croissance $\xrightarrow{2}$ phycomyces

("croissance" et "lumière" étant déjà reliés dans un syntagme de type consécutif croissance $\xleftarrow{3}$ lumière, il reste à trouver une liaison pour "phycomyces"). La première avec "lumière" n'est pas attestée dans le réseau notionnel et de ce fait est interdite tandis que la deuxième, valide, est acceptée :

croissance $\xleftarrow{2}$ phycomyces

b) La vérification sémantique est également nécessaire dans les cas où une relation entre deux descripteurs étant sûre d'un point de vue syntaxique, son orientation reste indéterminée. Entre un nom sujet par exemple et son complément, une relation associative ne fait pas de doute ; cependant elle ne s'établit pas toujours dans le même sens.

Ex. : La réaction de l'animal....
La réaction de peur....

7 - Résultats obtenus et leur synthèse

A l'issue de l'analyse syntaxique d'un texte et de l'application des règles de construction des syntagmes, les résultats obtenus s'expriment sous des formes diverses.

7.1 - Type de résultats

7.11 -Les syntagmes complets reliés ou non entre eux sous forme de graphes. Ils représentent le type de construction visée par les procédures décrites jusqu'ici.

7.12 -Les syntagmes incomplets composés seulement d'un descripteur et d'une relation spécifiée et orientée.

a) Ces syntagmes incomplets peuvent être dus à l'absence de l'un des correlands dans l'unité contextuelle. C'est une absence que l'on note parfois dans les titres des textes analysés.

Ex. : "Etude des effets de la réserpine".

En général un tel titre est complété par une phrase du type :

Ex. : "Effets sur le comportement du chat..."

Ces deux phrases séparées par un signe de ponctuation forte sont analysées séparément et l'on obtient ainsi : réserpine $\xrightarrow{3}$

et $\xrightarrow{3}$ comportement.

b) Les syntagmes incomplets sont également dus à la présence dans la phrase d'un terme naturel n'ayant pas d'équivalence en SYNTOL et cependant mis en relation avec un descripteur.

Ex. : La stimulation provoque un ensemble de troubles.

La relation consécutive s'établit entre "stimulation" et "ensemble". Le second mot n'ayant pas d'équivalent documentaire, on obtient : stimulation $\xrightarrow{3}$

c) De même la présence d'un pronom engendre la construction d'un syntagme incomplet. Il reprend en général un nom contenu dans une unité contextuelle différente et par conséquent non accessible en cours d'analyse.

Ex. : "Effets de la réserpine. Elle provoque des troubles".
"Etude de la réserpine. Son effet sur le comportement".

Dans les deux cas, la deuxième phrase ne contient pas l'agent de l'action ; on obtient :

$\xrightarrow{3}$ troubles
$\xrightarrow{3}$ comportement.

d) Enfin les syntagmes incomplets peuvent être dus à une caractéristique du mot-outil qui a servi à leur construction. On a vu par exemple que certains mots-outils comme les prépositions engendrent des syntagmes à un pôle.

7.13 -Les descripteurs isolés représentent les derniers types de résultats. Aucune relation ne les lie aux autres descripteurs de l'unité contextuelle, soit qu'aucun mot-outil ne les ait fait entrer dans un syntagme, soit que le code grammatical qui leur est associé n'ait pu être combiné suivant les règles d'une construction permise.

7.2 - Opérations de raccordement

Etant admis le principe que le graphe représentant le texte analysé doit être connexe, il faut relier entre eux les différents graphes obtenus après analyse fragmentée du texte. S'il est vrai que parfois les textes contiennent réellement des éléments hétérogènes et qu'on ne puisse alors réduire totalement les isolats, dans la plupart des cas on peut retrouver des corrélations que la limitation du contexte n'avait pas permis d'établir.

a) Certains enchaînements deviennent explicites grâce à la réduction à une seule, des occurrences des différents descripteurs.

b) D'autre part, des règles de raccordement peuvent s'appliquer à des syntagmes incomplets complémentaires. Par exemple, si deux descripteurs sont susceptibles d'être les pôles d'une même relation exprimée deux fois, il peut y avoir réduction de la relation et ainsi raccordement des descripteurs

$$\xrightarrow{3} B \quad \text{et} \quad A \xrightarrow{3} \quad \text{devient} \quad A \xrightarrow{3} B$$

Une vérification sémantique est toutefois nécessaire pour assurer la validité de ce raccordement.

c) D'autres règles de raccordement entre syntagmes ou graphes partiels isolés ont été proposées dans un précédent travail (1) ; elles seront également utilisées dans le cadre de cette expérience. Elles sont basées sur les affinités sémantiques et syntagmatiques entre descripteurs, affinités attestées les premières par le lexique documentaire, les secondes par le réseau notionnel.

---

(1) Voir bibliographie nº 5, Tome 4 : "L'Analyse Automatique".

Ex. : les deux syntagmes isolés $A \xrightarrow{3} B$ et $C \xrightarrow{3} D$ peuvent être associés dans un graphe

$$\begin{tikzcd} A \arrow[r, "3"] \arrow[d, "2"'] & B \\ C \arrow[r, "3"] & D \end{tikzcd}$$

si le réseau notionnel atteste l'existence d'une relation associative entre A et C, ou si ces deux descripteurs appartiennent à la même classe sémantique dans le lexique.

8 - CONCLUSION

Une des caractéristiques de la démarche qui vient d'être décrite est la place qu'y occupent respectivement les données syntaxiques et sémantiques : les différents outils qui ont été définis concourent en effet à n'utiliser des informations grammaticales que dans le cadre relativement restreint où elles sont sûres, et à les amalgamer avec des informations sémantiques provenant du réseau notionnel.

La préparation pratique d'une nouvelle expérience sur calculateur est en cours, qui déterminera la validité et les limites de cette méthode.

BIBLIOGRAPHY

Abraham S.Some questions of phrase structure
Grammars, I, CL IV, 1965

Abraham S. : An integrated language theory (to appear at Mouton & CO.)

Abraham S. : A theory of strutural semantics

Kiefer F. : Mouton & Co, The Hague-Paris, 1966

Chomsky N. : aspects of the theory of syntax, M.I.T. Press, Cambr., Mass., 1965

Katz, J.J. : The structure of a semantic Theory

Fodor, J.A. : Language,39, 1963

Katz, J.J. : An integrated theory of linguistic

Postal P.M. : Description, Cambr. Mass, 1964

1967 International Conference on Computational Linguistics

Axiomatic Characterization of Synonymy and Antonymy

H. P. Edmundson
University of California, Los Angeles

1. Introduction

1.1. Background

This work is a continuation of research reported in the paper Mathematical Models of Synonymy, which was presented at the 1965 International Conference on Computational Linguistics. That paper presented a historical summary of the concepts of synonymy and antonymy. It was noted that since the first book on English synonyms, which appeared in the second half of the 18th century, dictionaries of synonyms and antonyms have varied according to the particular explicit definitions of "synonym" and "antonym" that were used. The roles of part-of-speech, context of a word, and substitutability in the same context were discussed.

Traditionally, synonymy has been regarded as a binary relation between two words. Graphs of these binary relations were drawn for several sets of words based on Webster's Dictionary of Synonyms and matrices for these graphs were exhibited as an equivalent representation. These empirical results showed that the concepts of synonymy and antonymy required the use of ternary relations between two words in a specified sense rather than simply a binary relation between two words. The synonymy relation was then defined implicitly, rather than explicitly, by three axioms stating the properties of being reflexive, symmetric, and transitive. The antonymy relation was also defined by three axioms stating the properties of being irreflexive, symmetric, and antitransitive (the last term was coined for that study). It was noted that these six axioms could be expressed in the calculus of relations and that this relation algebra could be used to produce shorter proofs of theorems. However, no proofs were given. In addition, several geometrical and topological models of synonymy and antonymy were posed and examined.

It was noted that certain of these models were of more theoretical than practical interest. Each model was seen to be simple in that it could be expressed from mathematically elementary concepts, and each stressed certain aspects of the linguistic object being modeled at the expense of others. However, there seemed to be little theoretical preference among them. Their adequacy as models could be measured by their generality and predictive power. In terms of these criteria the algebraic model, whether expressed in terms of relations, graphs, or matrices, seemed to have the most usefulness. In part, this was due to the fact that one geometrical model, although highly suggestive, did not include a precise specification of the origin, axes, or coordinates for words in an n-dimensional space. Similarly, one topological model required a closure operation for each of the intensions or senses and had no linguistically interesting interpretation.

1.2 Summary

The present paper investigates more thoroughly the characterizations of synonymy and antonymy initiated in Edmundson (1965). In section 2, synonymy and antonymy are defined jointly and implicitly by a set of axioms rather than separately as before. First, it is noted that the original six axioms are insufficient to permit the proofs of certain theorems whose truth is strongly suggested by intuitive notions about synonyms and antonyms. In addition, it is discovered that certain fundamental assumptions about synonymy and antonymy must be made explicit as axioms. Some of these have to do with specifying the domain and range of the synonymy and antonymy relations. This is related to questions about whether function words, which linguistically belong to closed classes, should have synonyms and antonyms and whether content words, which linguistically belong to open classes, must have synonyms and antonyms. Several fundamental theorems of this axiom system are stated and proved. The informal interpretation of many of these theorems are intuitively satisfying. For example, it is proved that any even power of the antonymy relation is the synonymy relation, while any odd power is the antonymy relation.

In section 3, topological characterizations are posed and examined. A neighborhood topology is introduced by defining the neighborhood of a word. It is proved that this definition satisfies four neighborhood axioms. Also, a closure topology is introduced by defining the closure of a set of words. It is proved that this definition satisfies the four closure axioms.

2. Algebraic Characterization

2.1. Introduction - Relations

Before investigating antonymy and synonymy, we will establish some notions and notations for the calculus of binary relations.

Consider a set V of arbitrary elements, which will be called the universal set. A binary relation on V is defined as a set R of ordered pairs $\langle x,y \rangle$, where $x,y \in V$. The proposition that x stands in relation R to y will be denoted by xRy. The domain $\mathcal{D}(R)$, range $\mathcal{R}(R)$, and field $\mathcal{F}(F)$ of relation R are, respectively, defined by the sets

$$\{x:(\exists y)(xRy)\} \quad ; \quad \{y:(\exists x)(xRy)\} \quad ; \quad \{x:(\exists y)(xRy)\} \cup \{y:(\exists x)(xRy)\}$$

The complement, union, intersection, and converse relations are defined by

$$x\bar{R}y \equiv \sim xRy \quad ; \quad x(R \cup S)y \equiv xRy \vee xSy \quad ; \quad x(R \cap S)y \equiv xRy \wedge xSy \ ;$$
$$xR^{-1}y \equiv yRx$$

The identity relation I and null relation $\emptyset$ are defined by

$$xIy \equiv x = y \quad ; \quad x\emptyset y \equiv (x \neq x) \vee (y \neq y)$$

The product and power relations are defined by

$$xR|Sy \equiv (\exists z)[xRz \wedge zSy] \quad ; \quad R^n \equiv R|R^{n-1} \quad n \geq 1$$

Inclusion and equality of relations are defined by

$$R \subseteq S \equiv xRy \Longrightarrow xSy \quad ; \quad R = S \equiv R \subseteq S \wedge S \subseteq R$$

Later we will use the following elementary theorems which are stated here without proof:

Theorem: $R \subseteq S \implies R^{-1} \subseteq S^{-1}$

Theorem: $R \subseteq S \implies \overline{S} \subseteq \overline{R}$

Theorem: $(R^{-1})^{-1} = R$

Theorem: $(R|S)|T = R|(S|T)$

Theorem: $(R|S)^{-1} = S^{-1}|R^{-1}$

Theorem: $I|R = R|I = R$

Theorem: $S \subseteq T \implies R|S \subseteq R|T \wedge S|R \subseteq T|R$

2.2 Axioms and Definitions

Under the assumption that synonymy and antonymy are ternary relations on the set C of all content words, the following definitions will be used:

xS_iy ≡ word x is a synonym of word y with respect to the intension i (or word x is synonymous in sense i to word y)

xA_iy ≡ word x is an antonym of word y with respect to the intension i (or word x is antonymous in sense i to word y)

We will assume that the synonymy and antonymy relations are defined jointly and implicitly by the following set of axioms rather than separately as in Edmundson (1965).

Axiom 1 (Reflexive): $(\forall x)[xS_ix]$

Axiom 2 (Symmetric): $(\forall x)(\forall y)[xS_iy \implies xS_i^{-1}y]$

Axiom 3 (Transitive): $(\forall x)(\forall y)(\forall z)[xS_iy \wedge yS_iz \implies xS_iz]$

Axiom 4 (Irreflexive): $(\forall x)[x\overline{A}_ix]$

Axiom 5 (Symmetric): $(\forall x)(\forall y)[xA_iy \implies xA_i^{-1}y]$

Axiom 6 (Antitransitive): $(\forall x)(\forall y)(\forall z)[xA_iy \wedge yA_iz \implies xS_iz]$

Axiom 7 (Right-identity): $(\forall x)(\forall y)(\forall z)[xA_iy \wedge yS_iz \implies xA_iz]$

Axiom 8 (Nonempty): $(\forall y)(\exists x)[xA_iy]$

The properties named in Axioms 6 and 7 were coined for this study.

The above eight axioms may be expressed in the calculus of relations as follows:

Axiom 1 (Reflexive): $I \subseteq S_i$

Axiom 2 (Symmetric): $S_i \subseteq S_i^{-1}$

Axiom 3 (Transitive): $S_i^2 \subseteq S_i$

Axiom 4 (Irreflexive): $I \subseteq \overline{A}_i$

Axiom 5 (Symmetric): $A_i \subseteq A_i^{-1}$

Axiom 6 (Antitransitive): $A_i^2 \subseteq S_i$

Axiom 7 (Right-identity): $A_i|S_i \subseteq A_i$

Axiom 8 (Nonempty): $(\forall y)[A(y) \neq \emptyset]$ where $A(y) \equiv \{\langle x,y\rangle : x \in \mathcal{D}(A)\}$

This relation algebra will be used to produce shorter proofs, although this is not necessary. The consistency of this set of axioms is shown by exhibiting a model for them; their independence will not be treated.

In addition to the synonymy and antonymy relations it will be useful to introduce the following classes that are the images of these relations. The synonym class of a word y is defined by

$$s_i(y) \quad \equiv \quad \{x : xS_iy\}$$

which may be extended to an arbitrary set E of words by

$$s_i(E) \quad \equiv \quad \{x : (\exists y)[y \in E \wedge xS_iy]\}$$

Similarly, the antonym class of a word y is defined by

$$a_i(y) \quad \equiv \quad \{x : xA_iy\}$$

which may be extended to a set E of words by

$$a_i(E) \quad \equiv \quad \{x : (\exists y)[y \in E \wedge xA_iy]\}$$

2.3 Theorems

For reasons of notational simplicity, the subscript denoting the intension i will be omitted in the sequel whenever possible. However, the theorems must be understood as if the subscript were present.

As with any symmetric relation, it is possible to get stronger results than Axiom 2 and Axiom 5.

Theorem: $S^{-1} = S$

Proof: $S \subseteq S^{-1}$ by Axiom 2. Hence $S^{-1} \subseteq (S^{-1})^{-1} = S$. Therefore $S^{-1} = S$ by definition of equality.

Theorem: $A^{-1} = A$

Proof: Same as above theorem using Axiom 5.

Also we get a stronger result than the transitivity property of Axiom 3:

Theorem: $S^2 = S$

Proof: $S^2 \subseteq S$ by Axiom 3. Hence $S = S|I \subseteq S|S = S^2$ by Axiom 1. Therefore $S^2 = S$ by definition of equality.

In fact, by induction we have the generalization:

Theorem: $S^n = S \quad n \neq 0$

Proof: $S^n = S|S^{n-1} = S|(S|S^{n-2}) = \cdots = S|(S|(S|\cdots|S)\cdots) = S$.

It can be shown that antonymy and synonymy are distinct: $A \neq S$. In fact we have the stronger result:

Theorem: $A \subseteq \overline{S}$

Proof: Assume $A \nsubseteq \overline{S}$. Hence $A \cap S \neq \emptyset$ or $(\exists x)(\exists y)[x(A \cap S)y]$. Then $xAy \wedge xSy$ implies $xAy \wedge ySx$ by Axiom 2. So xAx, which contradicts $x\overline{A}x$ by Axiom 4: $I \subseteq \overline{A}$. Therefore $A \subseteq \overline{S}$.

Only because of Axiom 8, can we get a stronger result than the anti-transitivity property of Axiom 6.

Theorem: $A^2 = S$

Proof: $A \supseteq A|S$ by Axiom 7. Hence $A^2 = A|A \supseteq A|(A|S) = A^{-1}|(A|S) = (A^{-1}|A)|S$ since $A^{-1} = A$. Now $(\forall y)(\exists x)[xAy]$ by Axiom 8. So $(\forall y)(\exists x)[yA^{-1}x \wedge xAy]$ by Axiom 5. Hence $(\forall y)[yIy \Longrightarrow yA^{-1}|Ay]$. Thus $I \subseteq A^{-1}|A$. So $A^2 \supseteq I|S = S$. Therefore $A^2 = S$ since $A^2 \subseteq S$ by Axiom 6 and $S \subseteq A^2$.

The right-identity property of Axiom 7 can be strengthened to:

Theorem: $A|S = A$

Proof: $A|S \subseteq A$ by Axiom 7. Now $A = A|I \subseteq A|S$ since $I \subseteq S$. Therefore $A|S = A$ by definition of equality.

As a corollary we get that S and A commute:

Corollary: $A|S = S|A$

Proof: $A|S = A = A^{-1} = (A|S)^{-1} = (A^{-1}|S^{-1})^{-1} = S|A$

From the above two theorems it follows that:

Theorem: $S|A = A$

Proof: $S|A = A|S = A$.

As a special case we get:

Theorem: $A^3 = A|A^2 = A|S = A$.

In fact, we have the generalization:

Theorem: $A^n = \begin{cases} S \text{ if } n \text{ even} \\ A \text{ if } n \text{ odd} \end{cases}$

Proof: For n even, $A^n = A^{2k} = (A^2)^k = S^k = S$. For n odd, $A^n = A^{2k+1} = A|(A^2)^k = A|S = A$.

Next, several theorems about synonym classes and antonym classes will be stated and proved. First, the synonym class of a word is not empty:

Theorem: $s(y) \neq \emptyset$

Proof: Now $I \subseteq S$ by Axiom 1. So $(\forall y)[ySy]$. Hence $(\exists x)[xSy]$. Therefore, $s(y) \neq \emptyset$.

Because S is a symmetric relation, we have:

Theorem: $y \in s(x) \iff x \in s(y)$

Proof: $y \in s(x) \iff ySx \iff yS^{-1}x \iff xSy \iff x \in s(y)$.

Since S is reflexive, symmetric, and transitive, S is by definition an equivalence relation on the set C of all content words. Hence, we have the important result:

Theorem: $xSy \iff s(x) = s(y)$

Proof: ($\Longrightarrow$) Assume xSy. First let $u \in s(x)$. Then $uSx \wedge xSy \Longrightarrow uS^2y \Longrightarrow uSy \Longrightarrow u \in s(y)$. Hence $s(x) \subseteq s(y)$. Also $s(y) \subseteq s(x)$ by a similar argument. Therefore $s(x) = s(y)$.
($\Longleftarrow$) Assume $s(x) = s(y)$. Then $u \in s(x) \Longrightarrow u \in s(y)$. So $uSx \Longrightarrow uSy$. Hence $xSu \wedge uSy \Longrightarrow xS^2y \Longrightarrow xSy$. Therefore xSy.

In fact, we have the stronger result:

Theorem: $s(x) \cap s(y) = \begin{cases} s(x) & \text{if } xSy \\ \emptyset & \text{if } x\bar{S}y \end{cases}$

Hence for a given intension i the equivalence relation S_i partitions the set C of all content words into subsets that are disjoint (i.e., the subsets have no word in common) and exhaustive (i.e., every word is in some subset):

Theorem: $C = \bigcup_{x \in C} s_i(x)$

Second, the antonym class of a word is not empty:

Theorem: $a(y) \neq \emptyset$

Proof: Axiom 8: $(\forall y)(\exists x)[xAy]$ implies $a(y) \neq \emptyset$.

Note that a word does not belong to its antonym class:

Theorem: $y \notin a(y)$.

Proof: Assume $y \in a(y)$ so that yAy. But this contradicts Axiom 4: $yIy \implies y\bar{A}y$. Therefore $y \notin a(y)$.

Next we will establish some relations between synonym classes and antonym classes.

Theorem: $xAy \iff a(x) = s(y)$

Proof: ($\implies$) Assume $x \in a(y)$. First let $u \in a(x)$. Now $u \in a(x) \wedge xAy \implies uAx \wedge xAy \implies uA^2y \implies uSy \implies u \in s(y)$. Hence $a(x) \subseteq s(y)$. Also $s(y) \subseteq a(x)$ by a similar argument. Therefore $a(x) = s(y)$. ($\impliedby$) Assume $a(x) = s(y)$. But $y \in s(y) = a(x)$. Hence yAx. Therefore xAy by Axiom 5.

In fact, we get the following necessary and sufficient condition for equality:

Theorem: $a(x) = a(y) \iff s(x) = s(y)$

Proof: ($\implies$) Assume $a(x) = a(y)$. Now $a(x) \cap a(y) \neq \emptyset \implies (\exists z)[z \in a(x) \wedge a(y)] \implies (\exists z)[z \in a(x) \wedge z \in a(y)] \implies (\exists z)[zAx \wedge zAy] \implies (\exists z)[xAz \wedge zAy] \implies xA^2y \implies xSy$. Therefore $s(x) = s(y)$ by a previous theorem. ($\impliedby$) Assume $s(x) = s(y)$. Then xSy. First, let $u \in a(x)$. Then uAx. Hence $uAx \wedge xSy \implies uA|Sy \implies uAy \implies u \in a(y)$. Therefore $a(x) \subseteq a(y)$. Also $a(y) \subseteq a(x)$ by an identical argument. Therefore $a(x) = a(y)$.

2.4 Comments on the Algebraic Characterization

Even though $s(y) \neq \emptyset$ since ySy by Axiom 1, it may be necessary to add the following axiom:

$$\text{Axiom 9:} \qquad (\forall y)(\exists x)[x \neq y \wedge xSy]$$

to guarantee that the domain of the relation S is not trivial, i.e.,

$$s(y) - \{y\} \neq \emptyset$$

Axiom 9 is not necessary if $s(y)$ is permitted to be a unit set for certain words. Thus, we might define $s(y) = \{y\}$ for any function word y, e.g., s(and) = {and}. But this will not work for antonymy since $a(y)$ might be considered empty for certain words such as function words, e.g., a(and) = $\emptyset$. The alternative of defining $a(y) = \overline{\{y\}}$ is not reasonable since it produces more problems than it solves. Axiom 8: $(\forall y)(\exists x)[xAy]$ is reasonable if the contraries of words (e.g., nonuse, impossible, etc.) are permitted, i.e., $\bar{y} \in a(y)$.

The theorems

$$S^2 = S \quad , \quad A^2 = S \quad , \quad A|S = A \quad , \quad S|A = A$$

can be summarized in the following multiplication table for products of the relations S and A

	S	A
S	S	A
A	A	S

which is isomorphic to the table for addition modulo 2

	0	1
0	0	1
1	1	0

Note, even without Axioms 1-8, for

$$(1)\ A^2 = S \quad , \quad (2)\ A^3 = A \quad , \quad (3)\ A|S = A$$

that (1) and (2) imply (3), (1) and (3) imply (2), but (2) and (3) do not imply (1).

Suppose that for every pair <x,y> of words in the vocabulary V of a language exactly one of the following ternary relations holds:

(1) x and y are synonymous, xSy
(2) x and y are antonymous, xAy
(3) neither (1) nor (2), xMy

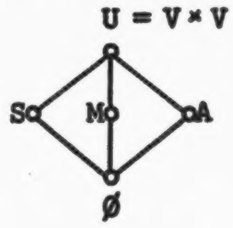
This can be expressed by

$$(Vx)(Vy)[x,y \in V \Longrightarrow xSy \vee xAy \vee xMy]$$

which is an exclusive disjunction. Thus the vocabulary V is partitioned as follows:

$$V = s(y) \cup a(y) \cup m(y)$$

for every word y. This also can be pictured in the lattice of relations



It can be shown that the multiplication table for products of the relations S,A, and M is

	S	A	M
S	S	A	M
A	A	S	M
M	M	M	M^2

3. Topological Characterizations

3.1. Introduction

We will now examine two topological models of synonymy. Being topological, they concern "semantic spaces" of words without any notion of "semantic distance" between two words. Again, we will restrict our attention to content words. Topological models for the antonymy relation will not be considered.

3.2. Neighborhood Topology

The first model considers a neighborhood topology, i.e., a topology based on neighborhoods. A set is said to have a neighborhood topology if there exist elements x called points and sets N_x called neighborhoods of x which satisfy the following axioms:

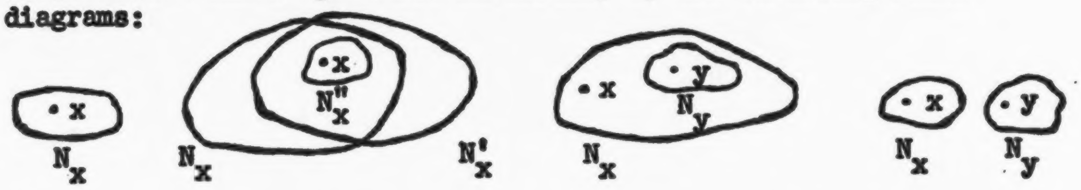
Axiom 1: $(\forall x)(\exists N_x)[x \in N_x]$

Axiom 2: $(\forall N_x)(\forall N'_x)(\exists N''_x)[N''_x \subseteq N_x \cap N'_x]$

Axiom 3: $(\forall y)(\forall N_x)(\exists N_y)[y \in N_x \implies N_y \subseteq N_x]$

Axiom 4: $(\forall x)(\forall y)(\exists N_x)(\exists N_y)[x \neq y \implies N_x \cap N_y = \emptyset]$

These axioms can be pictured informally by the following Euler diagrams:



Define a neighborhood $n_i(x)$ of a word x as any subset of the synonym class $s_i(x)$ of x that contains x, i.e.,

$$x \in n_i(x) \subseteq s_i(x)$$

Again, for reasons of notational simplicity, the subscript denoting the intension i will be omitted whenever possible.

First, neighborhood Axiom 1 is satisfied.

Theorem: $(\forall x)(\exists n(x))[x \in n(x)]$

Proof: By definition s(x) is a neighborhood n(x) of x containing x.

Second, neighborhood Axiom 2 is satisfied.

Theorem: $(\forall n(x))(\forall n'(x))(\exists n''(x))[n''(x) \subseteq n(x) \cap n'(x)]$

Proof: For arbitrary n(x) and n'(x), let $n''(x) = n(x) \cap n'(x)$. Then $n''(x) \subseteq s(x)$ since $n''(x) = n(x) \cap n'(x) \subseteq s(x) \cap s(x) = s(x)$. Also, $x \in n''(x)$ since $x \in n(x) \wedge x \in n'(x)$ imply $x \in n(x) \cap n'(x) = n''(x)$. Therefore, $(\forall n(x))(\forall n'(x))(\exists n''(x))[n''(x) \subseteq n(x) \cap n'(x)]$.

Third, neighborhood Axiom 3 is satisfied.

Theorem: $(\forall y)(\forall n(x))(\exists n(y))[y \in n(x) \implies n(y) \subseteq n(x)]$

Proof: For arbitrary $y \in n(x)$, let $n(y) = n(x)$. But $y \in n(x)$ implies $s(x) = s(y)$ since $y \in n(x) \subseteq s(x) = \{z : zSx\}$ implies ySx and ySx implies $s(y) = s(x)$. Then $n(y) \subseteq s(y)$ since $n(y) = n(x) \subseteq s(x) = s(y)$. Also $y \in n(y)$ since $y \in n(x) = n(y)$. Therefore, $(\forall y)(\forall n(x))(\exists n(y))[y \in n(x) \implies n(y) \subseteq n(x)]$.

In fact, the neighborhood topology satisfies Axiom 4, which is a separation axiom:

Theorem: $(\forall x)(\forall y)(\exists n(x))(\exists n(y))[x \neq y \implies n(x) \cap n(y) = \emptyset]$

Proof: Assume $x \neq y$. Let $n(x) = \{x\}$ and $n(y) = \{y\}$. Then $x \in n(x) \subseteq s(x)$ and $y \in n(y) \subseteq s(y)$. Thus $n(x) \cap n(y) = \{x\} \cap \{y\} = \emptyset$ since $x \neq y$.

Therefore, with respect to synonymy, words have a neighborhood topology since

(1) $(\forall x)(\exists n(x))[x \in n(x)]$
(2) $(\forall n(x))(\forall n'(x))(\exists n''(x))[n''(x) \subseteq n(x) \cap n'(x)]$
(3) $(\forall y)(\exists n(y))[y \in n(x) \implies n(y) \subseteq n(x)]$
(4) $(\forall x)(\forall y)(\exists n(x))(\exists n(y))[x \neq y \implies n(x) \cap n(y) = \emptyset]$

3.3. Closure Topology

The second model considers a closure topology, i.e., a topology based on a closure operation. A set is said to have a closure topology if there exists a unary operation on its subsets, denoted by $\sim$ and called the closure, which satisfies the following axioms:

Axiom 1: $\tilde{\emptyset} = \emptyset$
Axiom 2: $E \subseteq \tilde{E}$
Axiom 3: $\tilde{\tilde{E}} \subseteq E$
Axiom 4: $\widetilde{E \cup F} = \tilde{E} \cup \tilde{F}$

Define the closure of a set E of words as the synonym class of E, i.e.,

$$\tilde{E} \equiv s(E)$$

The closure axioms can be shown to be satisfied by using the original definition of synonym class

$$s(E) \equiv \{x : (\exists y)[y \in E \wedge xSy]\}$$

However, shorter proofs are possible by noting that the synonym class of a set E of words can be expressed as

$$s(E) = \bigcup_{y \in E} s(y) = \bigcup_{y \in E} \{x : xSy\}$$

First, closure Axiom 1 is satisfied:

Theorem: $s(\emptyset) = \emptyset$

Proof: $S(\emptyset) = \bigcup_{y \in \emptyset} s(y) = \emptyset$

Second, closure Axiom 2 is satisfied:

Theorem: $E \subseteq s(E)$

Proof: $s(E) = \bigcup_{y \in E} s(y) \supseteq \bigcup_{y \in E} \{y\} = E$ since $y \in s(y) \implies \{y\} \subseteq s(y)$.

Third, closure Axiom 3 is satisfied:

Theorem: $s[s(E)] \subseteq s(E)$

Proof: Now $s(s(y)) = s(\{u : uSy\}) = \{v : vS^2y\} \subseteq \{v : vSy\} = s(y)$ since $S^2 \subseteq S$. Thus

$$s[s(E)] = \bigcup_{x \in s(E)} s(x) \; . = \bigcup_{x \in \bigcup_{y \in E} s(y)} s(x) = \bigcup_{y \in E} \bigcup_{x \in s(y)} s(x) = \bigcup_{y \in E} s(s(y)) \subseteq \bigcup_{y \in E} s(y) = s(E)$$

Fourth, closure Axiom 4 is satisfied:

Theorem: $s(E \cup F) = s(E) \cup s(F)$

Proof: $$s(E \cup F) = \bigcup_{y \in E \cup F} s(y) = \bigcup_{y \in E} s(y) \cup \bigcup_{y \in F} s(y) = s(E) \cup S(F).$$

Therefore, with respect to synonymy, words have a closure topology since

(1) $s(\emptyset) = \emptyset$
(2) $E \subseteq s(E)$
(3) $s[s(E)] \subseteq s(E)$
(4) $s(E \cup F) = s(E) \cup s(F)$

Note that from Axioms 2 and 3 we get

Theorem: $s[s(E)] = s(E)$

3.4. Comments on Topological Characterizations

Note that for the neighborhood topology a separation axiom has been added to the three axioms proposed in Edmundson (1965). Also, the neighborhood topology seems more intuitively satisfying than the closure topology. However, for the closure topology if we define the derived set of a set E of words as the set of all words that are synonymous to some word of E, but not identical to that word. i.e.,

$$E' \equiv \{x : (\exists y)[y \in E \wedge x \neq y \wedge xSy]\}$$

then we have the following result:

Theorem: $s(E) = E \cup E'$

which may be given a reasonable linguistic interpretation. An example is $\{y\}' = s(y) - \{y\}$ which was discussed in the section on algebraic characterization.

4. Conclusions

These results support the belief that the algebraic characterization is insightful and appropriate. For example, the assumption that synonymy is an equivalence relation also has been made, either directly or indirectly, by F. Kiefer and S. Ábrahám (1965), U. Weinreich (1966), and others. Since the axiom system defines the notions of synonymy and antonymy jointly and implicitly, it avoids certain difficulties that are encountered when attempts are made to define these notions separately and explicitly.

These topological characterizations provide a nonmetric representation of what has been called informally a "semantic space". Previous attempts to construct a semantic space that is metric (i.e., one for which a distance function is defined) have not met with much success. The consideration of general topological spaces avoids this difficulty.

References

R. Carnap, Introduction to Symbolic Logic and Its Applications, W. Meyer and J. Wilkinson (trs.), Dover, N. Y., 1958.

H. P. Edmundson, "Mathematical Models of Synonymy", International Conference on Computational Linguistics, 1965.

P. Kiefer and S. Ábrahám, "Some Problems of Formalization in Linguistics", Linguistics, v. 17, Oct. 1965, pp. 11-20.

V. V. Martynov, Pytannja prikladnoji lingvistyky; tezisy dopovidej mižvuzovs'koji naukovoji konferenciji, Sept. 22-28, 1960, Černivcy.

A. Naess, "Synonymity as Revealed by Intuition", Philosophical Review, v. 66, 1957, pp. 87-93.

U. Weinreich, "Explorations in Semantic Theory", in Current Trends in Linguistics, III, T. Sebeok (ed.), Mouton and Co., The Hague, 1966.

P. Ziff, Semantic Analysis, Cornell University Press, Ithica, N. Y., 1960.

28

Semantically Governed Machine Translation of BE-clauses with Adverbs and Prepositional Phrases. Demonstration of a Ring-Model for German/English, English/German including Analysis-Synthesis, Transformation, Transition, and Generation.

Klaus Günther Schweisthal, Forschungsgruppe LIMAS, Bonn

Research in methods of lexicalization, classification, analysis-synthesis, transformation, transition, and generation, which has been carried on independently in the German and English sections at LIMAS, has recently been consolidated for the purpose of constructing a Ring-Translation-Model for German/English, English/German.

This model has to meet the following requirements:

1. It must produce a translation which is identical in meaning to the original and stylistically acceptable.
2. Some of the difficulties which have heretofore appeared in machine translation must be resolved.

The starting point (Figure 1) is a German adverbial phrase, which is to be analyzed and then converted into a factor formula by means of a morpho-nomo-transformation. An explicative nomo-nomo-transformation together with prepositional governing factors then expand the factor formula for the adverbial expression into a formula for a prepositional phrase, which in turn becomes expressible through a nomo-morpho-transformation. The factor formula for the German prepositional phrase is converted into an English factor formula according to the coordinator's system of maximum agreement. This takes place through a nomo-nomo-transition if the identical factor formula is present in English, otherwise through a nomo-nomo-transformation. A nomo-morpho-transformation then creates an expressible English prepositional phrase which is semantically equivalent to the German. The factor formula for an English adverbial phrase is made possible by a reductive nomo-nomo-transformation plus a nomo-morpho-transformation. The last step of the cycle again takes place in the coordinator. The factor formula for the English adverbial phrase is converted into the corresponding formula for the German adverbial phrase by means of transition or a nomo-nomo-transformation. The morphological realization of this factor formula is identical to that of the initial German adverbial phrase.

A sample output of the ring would be:

Vater ist drinnen ---- Vater ist innerhalb des Hauses ----
Father is inside (of) the house ---- Father is inside ----
Vater ist drinnen.

The automatic reversible process of the ring model is determined by several association lists and function matrices. At first a

Reversible Ringtransformation Deutsch Englisch

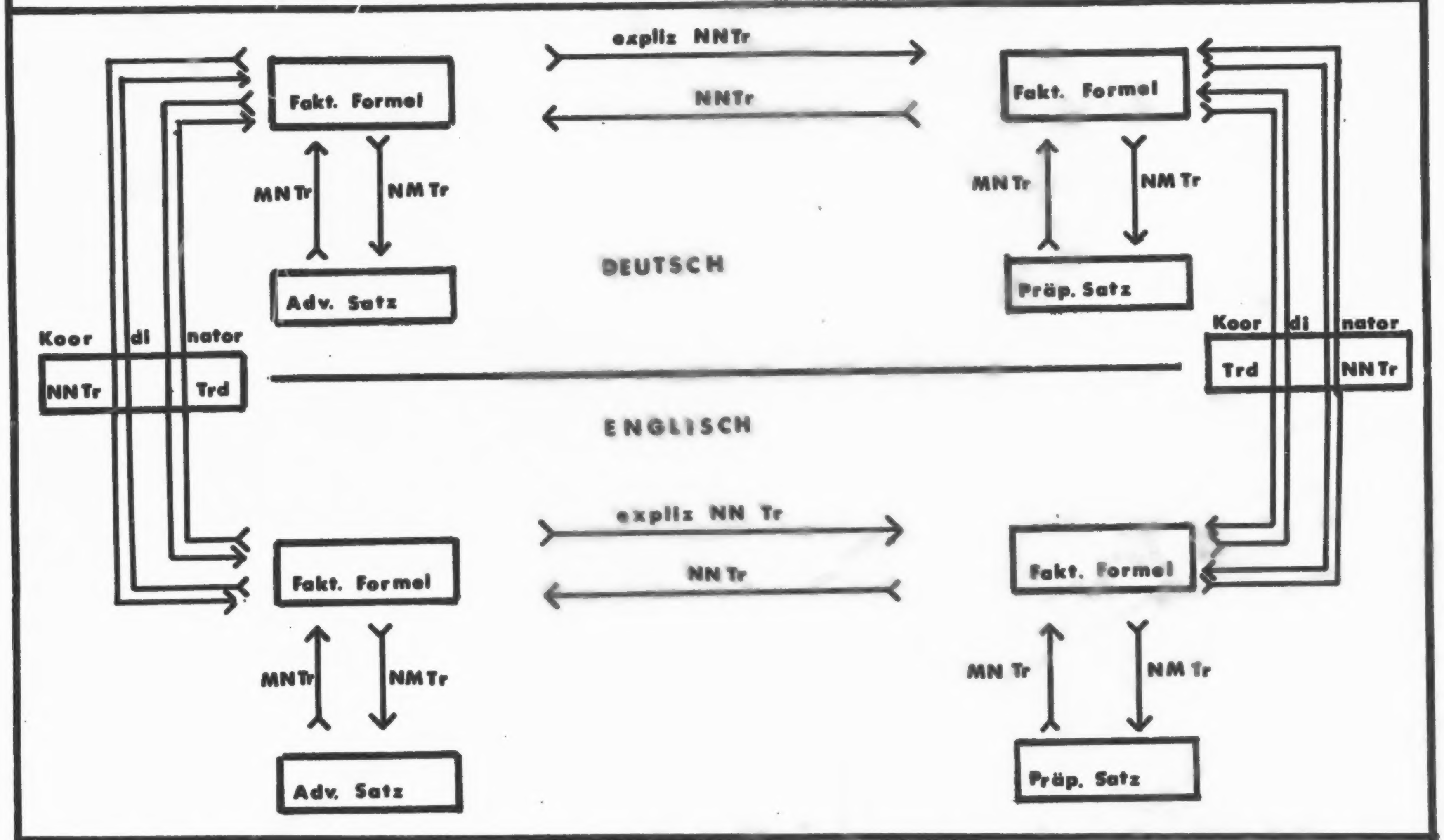

expliz NNTr
NNTr
Fakt. Formel
MNTr
NMTr
Adv. Satz
DEUTSCH
Fakt. Formel
MNTr
NMTr
Präp. Satz
Koor di nator
NNTr
Trd
Koor di nator
Trd
NNTr
ENGLISCH
expliz NN Tr
NN Tr
Fakt. Formel
MNTr
NMTr
Adv. Satz
Fakt. Formel
MN Tr
NM Tr
Präp. Satz

Figure1

Forschungsgruppe LIMAS 67

lexicon of German and English semantic factors is necessary.

This list forms the upper row for the necessary nomo-association lists of the German-English words (Figure 2). In this figure are illustrated three different types of semantic association.

List A/D illustrates the association of the factors to the German adverbs, A/P to the German prepositions, and N/D to the various German noun classes. In a strict sense the noun class association is to be specified once again according to the two syntactic functions of the noun in the sentence, i.e. BE-subject* and BE-complement*.

For English the association has been organized in a similar manner in the lists A/E, P/E, N/E, but in such a way that they are independent of German.

An example may make this a little more clear. Assume that A/D_1 is the German adverb "drinnen" and the factors 1, 3, 5, 8, → are coded symbols for the factors Ort, Lage, Einschluß, Betonung der einschließenden Grenzen, Hinweis, then according to the principle of maximum agreement we are able to determine the same factor combination Place, Location, Interior, Accent on boundaries, Reference, in A/E under 1, which in English is the word "inside".

With this same set of factors we find that in the lists P/D and P/E according to maximum agreement the results are P/D_1 and P/E_1, which are expressed as "innerhalb" and "inside of".

In the list N/D all five preposition factors appear in the noun class B, so that if we were to choose as BE-complements the words Garten or Haus the resultant phrase would be: innerhalb des Gartens (Hauses). Because this set of factors also appears under B in N/E, it is possible through transition to produce the English phrase "inside of the garden (house)".

In view of this schematic arrangement one might think that it is possible to associate all German prepositions word for word with English prepositions. (This method was used frequently in the early days of machine translation resulting in untold difficulties.) Such word for word association is almost 100 % possible for the prepositions of place.

But if, for example, we are dealing with the German word nach, which can either be P/D_2 plus factor 7 (Zeit) or P/D_4 plus factor 2 (Ort), we are not able to make a direct association, but must decide according to the method of maximum agreement whether English "after" or "to" is meant.

The corresponding words in A/D and A/E would be "danach"-"dorthin", and "after that"-"there".

---

* See accompanying sheet for LIMAS terminology

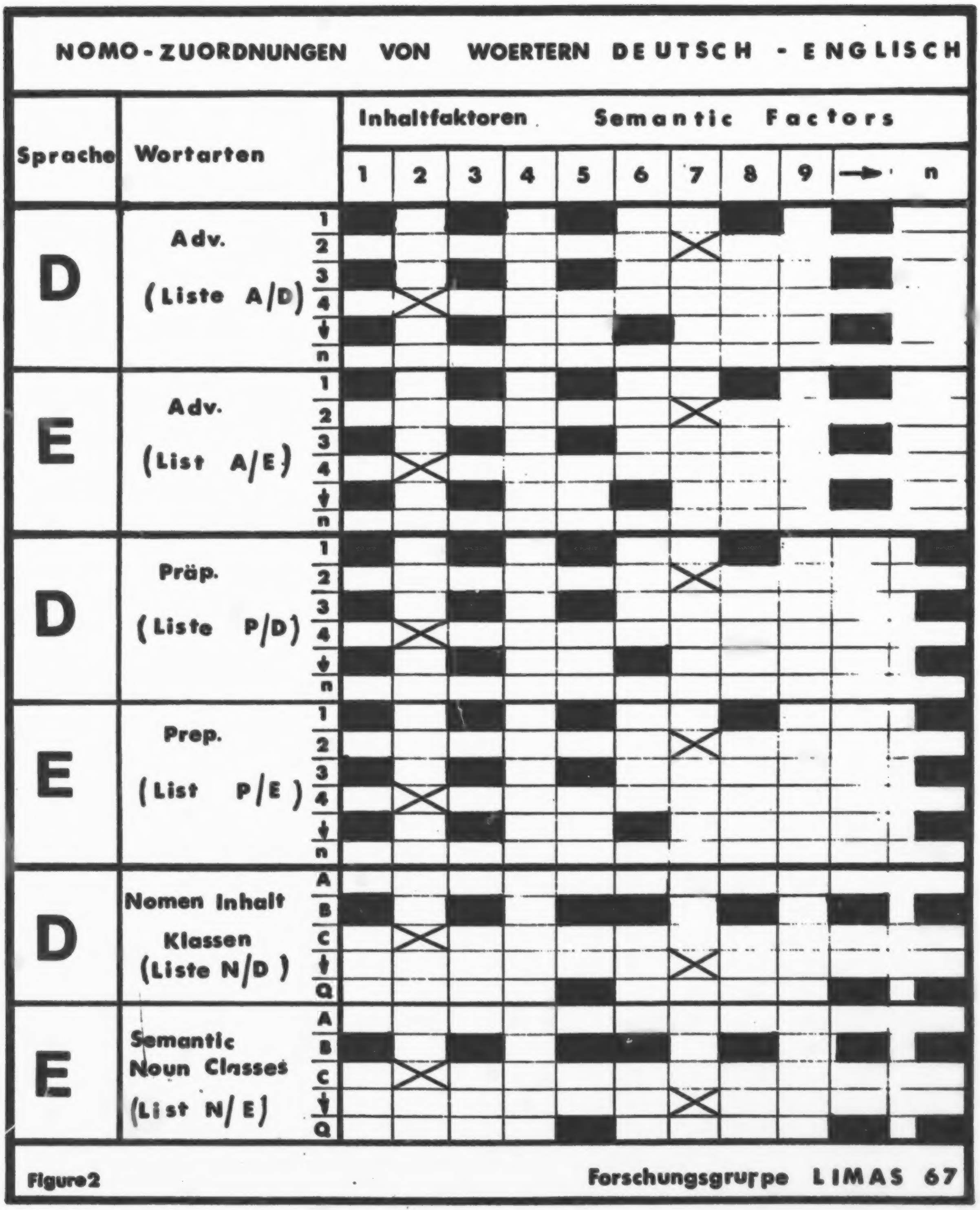

NOMO-ZUORDNUNGEN VON WOERTERN DEUTSCH - ENGLISCH
Sprache
Wortarten
Inhaltfaktoren. Semantic Factors
1
2
3
4
5
6
7
8
9
→
n
D
Adv.
(Liste A/D)
E
Adv.
(List A/E)
D
Präp.
(Liste P/D)
E
Prep.
(List P/E)
D
Nomen Inhalt Klassen
(Liste N/D)
E
Semantic Noun Classes
(List N/E)
1
2
3
4
↓
n
A
B
C
↓
Q

Figure2 Forschungsgruppe LIMAS 67

The selection of the factors Place and Time for the respective noun classes in the noun class lists N/D and N/E would allow the generation of the phrases: "nach dem Mittagessen", "nach Köln", and "after lunch", "to Cologne".

Already it is evident how our system differs from those of earlier machine translation projects.

In the nomo-association list we have established the prerequisites for generation, transformation and transition. In the middle function matrix P/D - P/E (Figure 3) of this diagram are contained the restrictions as to which prepositions may combine with which semantic noun classes to produce a meaningful Be-complement. A quick glance shows that the preposition-noun relationship is differentiated in various ways. This is one proof for the necessity of classifying nouns according to function or semantic classes.

This can be illustrated in the following example. The codes in the preposition column are replaced by prepositions, and the row of noun classes by a representative of each class.

	A-Vater	B-Garten	C-Köln	→Mittagessen	Q-Gedanke
1. innerhalb		x	x		
2. rechts	x	x	x		
4. nach (Zeit)				x	
n. vor (Abstr. Ber.)					x

Using the lexicon the permitted BE-complements can be programmed from this matrix.

The various possibilities for generating meaningful adverbial BE-clauses can be seen in the matrix N/D - A/D. Here again the necessity of using noun classes is evident, even though some noun classes in the matrix (A, B, C,) appear to be similar.

Example: Vater ist drinnen
draußen
rechts
but not danach

But with Mittagessen all four adverbs are possible:

drinnen
draußen
rechts
danach

A further prerequisite for the generation of meaningful BE-clauses containing prepositional phrases is the coordination of the BE-subject noun class with the BE-complement noun class.

The matrix Sntr/N/D - SnE/N/D supplies the necessary information.

NOMO - ZUORDNUNG VON SATZGLIEDERN DEUTSCH - ENGLISCH

DEUTSCH	MATRIX	ENGLISCH

ERLAUBTE ADVERBIALSAETZE

Matrix:

N/D — A/D

Adv. / Nomen		SnE 1	2	3	→	n
Sn tr	A	X	X	X		
	B	X	X	X		
	C	X	X	X		
	↓	X	X	X	X	
	Q					X

PERMITTED ADVERBIAL CLAUSES

Matrix:

N/E — A/E

ERLAUBTE PRAEPOS. PHRASEN

Matrix:

P/D — N/D

Seinsergänzung

P \ N	A	B	C	→	Q
1		X	X		
2	X	X	X		
↓				X	
n					X

Be– Complement

PERMITTED PREPOS. PHRASES

Matrix:

P/E — N/E

ERLAUBTE FUNKTIONEN

Sntr /N Inh. Klasse –

Sn E/N Inh. Klassen

Matrix:

Sntr/N/D–SnE/N/D

Sn tr \ Sn E	A	B	C	→	Q
A	X	X	X		
B	X	X	X		
C	X	X	X		
↓	X	X	X	X	
Q					X

PERMITTED FUNCTIONS

BeS/Sem. Noun Class –

BeC/Sem. Noun Classes

Matrix:

BeS/N/E –BeC/N/E

Figure3

Forschungsgruppe LIMAS 67

For example, a BE-subject from the noun class "abstract" is compatible only with a BE-complement from the same class.

Die Idee ist hinter dem Gedanken
but not Der Gedanke ist hinter dem Garten
Der Garten ist hinter dem Haus
but not Der Garten ist hinter dem Mittagessen

Not until all of these factors have been formalized and all conditions which were illustrated in the matrices are taken into consideration it is possible to set up a final function matrix. This matrix then aids the translation process in the generation of meaningful BE-clauses with prepositional phrases. (Figure 4)

For example:

A is to be replaced by Vater - (father)
B Haus - (house)
C Köln - (Cologne)
→ Mittagessen (action) - (lunch)
Q Gedanke - (thought)
9 vor (place) - (in front of)

then

NPN/D --> A 9 A --> Vater ist vor dem Vater
--> A 9 B --> vor dem Haus
--> A 9 C --> vor Köln
--> A 9 → (not permitted!) Vater ist vor dem Mittagessen
--> A 9 Q (not permitted!) Vater ist vor dem Gedanken

NPN/E --> A 9 A --> Father is in front of the father
--> A 9 B --> in front of the house
--> A 9 C --> in front of Cologne
--> A 9 → (not permitted!) Father is in front of lunch
--> A 9 Q (not permitted!) Father is in front of the thought

In order to show that this model exists not only on paper but also in practise, we have compiled a German and English lexicon consisting of 16 prepositions, 16 adverbs, and one noun from each of the five noun classes. The upper row of the nomo-association chart, which was seen in figure 2, is filled with over 30 semantic factors.

The outputs show that the ring model works satisfactorily.

With the aid of these matrices (Figures 1 - 4), a semantically equivalent German/English, English/German translation can be executed for all the cases in which the function matrices in both languages are identical in structure, e.g. (NPN/D = NPN/E ---- (transition).

For those cases in which the function matrices are not identi-

Erlaubte ist-Sätze mit Präp.-Phrasen

KOMPLEXE MATRIX

NPN/D

SnE / Sntr	A				B				C				→				Q			
A			9		1		9		1		9			⑤						
				14	2			14	2	⑥					⑩					
	3	7			3	7			3	7						⑮				
	4				4		12		4		12									
B			9		1		9		1		9									
				14		2		14	2											
	3	7			3	7			3	7										
	4				4		12		4		12									
C			9		1		9		1		9									
				14	2				2											
	3	7			3	7			3	7										
	4				4		12		4		12									
↓			9		1		9		1		9			5						
				14	2			14	2						10					
	3	7			3	7			3	7						15				
	4				4				4											
Q																	1a			13
																	2a			
																			11	
																		8		16

○ nur für "geht"

Permitted is-clauses containing prep.-phrases

COMPLEX MATRIX

NPN/E

Figure 4

cal ($NPN/D \neq NPN/E$), the proper translation is insured by means of an auxiliary program. This takes place through *verbalization* of the semantic factors of the source language in the target language.

For example: The English sentence *The candle is inside (of) the father* cannot be translated into German as it now is because there is no corresponding matrix NPN/D. The verbalization program, however, makes possible the translation *Die Kerze ist im Körper des Vaters eingeschlossen*.

If a German adverb has no English counterpart, an auxiliary program will create a semantically equivalent prepositional phrase, e.g. *dahinter* ---- *behind it*. All German adverbs must be provided with the possibility of such a *transformation*, so that in the reverse process - in case it should be necessary - the proper German adverb can be selected for the English combination preposition + pronoun.

Some Terms of LIMAS-Terminology

1. Transformation

a) Morpho-transformation: A change within the morpho-structure involving formal constituents.

b) Morpho-semantic-transformation: A change within the morpho-level involving those semantic factors which are referred to as "Denkweisen". These are: sentence constituent,framed, stressed, cyclopic, nominal, actional, and designative.

c) Morpho-nomo, and nomo-morpho-transformation: A transfer from the morpho-level to the nomo-level or vice versa. (This is similar to a change from the deep structure to the surface structure and vice versa in Transformational-Generative Grammar.)

d) Nomo-nomo-transformation:
  1) Within the same language: A change of the factor formula by means of exchange, addition, or elimination of certain factors.
  2) Between the source language and the target language: The factor formula of the source language is not acceptable in the target language, and must therefore be reshaped.

2. Transition

The simple transfer of a factor formula from one metalanguage into another.

3. Explication

To make explicit a semantic factor in a sentence which contains no morpheme to represent that factor. The morpheme can be lacking for three reasons:

a) ellipsis

b) non-expressed context relations

c) a combination of semantic factors which are understood but not expressed.

It is therefore important to differentiate between a) ellipsis explication, b) context-relation explication, and c) factor explication. These three types of explication can, however, be combined.

4. Verbalization

The process of verbally expressing a semantic factor within the morpho-level which has been determined through explication.

5. **Reduction**

The opposite of explication, i.e. a factor formula of the metalanguage is not permitted to be verbalized. The factor is not really eliminated, but is implied within the morpho-level.

6. **Explicating-transformation**

   a) Explicating nomo-nomo-transformation: A semantic factor which has been determined through explication changes the factor formula.

   b) Explicating nomo-morpho-transformation: The semantic factor becomes verbalized.

Faktorenformel	-	factor formula
Funktionsmatrix	-	function matrix
Inhaltfaktor	-	semantic factor
ist-Satz	-	Be-clause
Nomeninhaltklasse	-	semantic noun class
Seinsträger (Sntr)	-	Be-subject (BeS)
Seinsergänzung (SnE)	-	Be-complement (BeC)
Traduktion (Trd)	-	transition
Transformation (Tr)	-	transformation
Zuordnungsliste	-	association list

Criteria for "meaningful" BE-clauses with adverbs and prepositional phrases:

a) Approximately 40,000 examples of modern German taken from various sources.

b) A smaller collection of English examples.

c) Interviews with several native speakers of English from England and America.

29

L'explication automatique du facteur sémantique 'appartenance' apparaissant dans le "Zuwendsatz" [+] allemand et la traduction automatique de ce facteur: de l'allemand vers l'anglais et de l'anglais vers l'allemand.

Dr. Alfred Hoppe, Forschungsgruppe LIMAS, Bonn

Objectifs:

On va montrer, en prenant l'exemple de la proposition allemande d'attribution

1°) comment un facteur sémantique contenu dans un modèle syntaxique de la langue de départ, sans qu'on puisse le faire voir sur le plan morphologique, sera trouvé mécaniquement (explication);

2°) comment ce facteur sera utilisé à l'occasion de la traduction;

3°) comment, à cette occasion, un modèle syntaxique de la langue de départ sera transformé en un autre, qui appartient à la langue d'arrivée;

4°) - si l'on suppose que la langue d'arrivée ne dispose pas du modèle syntaxique figurant dans la langue de départ -;

5°) comment le facteur trouvé apparaîtra, dans la langue d'arrivée, sur le plan morphologique du modèle syntaxique;

6°) comment ce système fonctionnera d'une façon réversible.

[+] Le "Zuwendsatz" fait partie des modèles syntaxiques ou schémas syntaxiques ("Satzpläne") dont dispose une langue. Il s'agit, en l'occurrence, d'un plan de construction de propositions typiquement allemandes, p.ex.:"Er trat seinem Freund auf den Fuß", où le complément indirect allemand se rend en français par un complément d'appartenance: "sur le pied de son ami". Des états, événements, activités ou actions localisés sont nécessairement tournés vers quelqu'un (voir Grammaire DUDEN, éd. 1959, p. [466] seq.). Il s'agit donc d'une espèce de tropisme (tourner vers). Dans la suite, j'appellerai les réalisations de ces modèles syntaxiques "propositions d'attribution", et vous voudrez bien entendre l'attribution dans ce sens étroit de tropisme.

Soient les exemples suivants:

a) The nurse laid her hand on the patient's shoulder -
Die Schwester legte dem Patienten die Hand auf die Schulter
(L'infirmière posa la main sur l'épaule du malade).

b) The nurse sent the newspaper to the patient at home -
Die Schwester schickte dem Patienten die Zeitung ins Haus
(L'infirmière expédia le journal au malade, chez lui).

Proposition a): la main appartient à l'infirmière (appartenance);
l'épaule, au malade (appartenance).

Proposition b): le journal, respectivement la maison, n'appartient
ni à l'infirmière ni au malade.

Remarques préalables.

A. Terminologie

L'attribution est un facteur de contenu: un sujet devient actif et son activité se dirige vers une grandeur, normalement personne, sans que cette grandeur soit l'objet de l'activité; elle "participe plutôt à l'activité" en tant que celle-ci est tournée vers elle (L. Weisgerber, loc. cit. p. 276).
On appellera cette espèce d'activité "attribution", et celui qui participe à l'activité, "grandeur d'attribution". Une structure de contenu couvrant l'attribution à une telle grandeur s'appellera plan d'attribution.
Il se compose:
1°) d'un sujet agissant,
2°) d'une attribution,
3°) d'une grandeur d'attribution.
En allemand, cette grandeur apparaît au cas - objet indirect ou secondaire (Dativ), p.ex.: "Die Schwester hilft dem Patienten." (En allemand! Par contre en français: "L'infirmière secourt le malade.")

Le plan d'attribution pourra se modifier par extension. Ces modifications dépendront de la rection du verbe et pourront consister en:

1°) un objectif d'activité. Il s'agit de l'objet visé par l'activité vue sous l'angle de l'attribution;

2°) un objectif de relation. Il s'agit de la grandeur rattachée à l'activité par une relation déterminée (lien prépositionnel fixe du verbe: "antworten auf die Frage" - "répondre à la question");

3°) une orientation ou situation spatiale liée, sur le plan du contenu, à l'activité: "legen - auf die Schulter" - "poser sur l'épaule", "falten - auf der Brust" - "joindre sur la poitrine";

4°) une indication de l'instrument servant à l'activité: "drohte mit dem Finger" - "menaçait de son doigt";

5°) une identification avec une caractéristique: "behilflich" - "secourable", ou une grandeur: "eine Mutter" - "une mère": Die Schwester war dem Patienten behilflich - eine Mutter.

Ces modificateurs exigés par la rection du verbe sont à considérer comme des cases vides sémantiques qui, sur le plan morphologique, pourront figurer comme complément prépositionnel, complément d'objet, "phrase" [+] prépositionnelle ou attribut, pourvu qu'on soit en présence d'un modèle syntaxique permettant l'emploi de ces formes. Un autre modèle pourra demander d'autres formes.

Ces modificateurs, dépendant de verbes, du plan d'attribution constituent des éléments nécessaires du complexe de contenu; à leur défaut, aucun des plans syntaxiques dans lesquels ils pourraient figurer morphologiquement parlant ne serait considéré comme une réalisation complète. Etant donc indispensables à la constitution des propositions, ils seront appelés constituants de propositions.

A côté de ces modificateurs de la proposition d'attribution conduisant à des extensions et dépendant du verbe, il y a un autre

---

[+] Le terme "phrase" sera pris dans le sens anglais: un groupe de mots sans sujet ni prédicat, et sera prononcé à l'anglaise.

groupe de modificateurs: ceux-ci ne sont pas régis par le verbe mais ne sauraient figurer que dans le cadre des cases vides établies par la rection. Ce sont les classes de contenus lexématiques: c'est ainsi que la grandeur d'attribution, par exemple, sera représentée normalement par un nom de personne, tandis que les autres extensions pourront être des noms désignant une partie du corps ou non.

Les modificateurs régis par le verbe et les classes de contenus lexématiques qui se présentent dans le cadre de ceux-là donneront en tout 34 modifications différentes apportées au plan d'attribution. Ces modifications seront appelées modèles de plan d'attribution. Même s'il n'y a pas de modificateur, le plan d'attribution lui-même comptera dans le nombre des modèles de plan d'attribution.

Parmi les quelque 4oo verbes allemands régissant ce qu'on appelle en allemand un "Dativ", il y en a dont le régime, membre d'attribution, ne représente pas un constituant nécessaire de proposition. A défaut de celui-ci, il n'y aura pas plan d'attribution et ce cas ne nous intéressera pas ici.

La réalisation morphologique des 34 modèles de plan d'attribution conduira à des propositions ou des "phrases" d'attribution (attributive Phrasen). Celles-ci seront considérées comme des patterns de propositions ou de "phrases" en tant qu'elles représenteront des patterns formels pouvant également contenir d'autres complexes de contenu dès que l'on fera abstraction, dans le complexe de contenu, du facteur grandeur d'attribution; c'est-à-dire, le pattern de proposition ou de "phrase" ne permet une proposition ou une "phrase" d'attribution que dans le cas où un modèle de plan d'attribution s'y réalisera.

Ayant donc compté 21 patterns de propositions ou de "phrases" dont chacun est susceptible de réaliser l'ensemble des 34 modèles de plan d'attribution, nous aurons en tout 714 propositions ou "phrases" d'attribution différentes.

Il importe encore de relever le fait que le plan d'attribution et les modèles de ce plan font partie de la nomosphère, tandis que les propositions et les "phrases" d'attribution, ainsi que les patterns de propositions ou de "phrases" servant à les réaliser appartiennent à la morphosphère [+].

D'après tout ce que je viens de dire, tirons pour le moment les conséquences suivantes, qui ont leur importance:

1°) le plan d'attribution ne saurait, sur le plan du contenu, apparaître que dans un quelconque de ses 34 modèles; il représente ce qui, sur ce plan même, est commun à tous les modèles;

2°) chacun de ces modèles pourra s'exprimer, linguistiquement parlant, dans chacun des 21 morphopatterns. Ni le plan d'attribution ni aucun modèle de plan d'attribution ne sauraient s'exprimer sans qu'on recoure à un morphopattern; cette expression requiert donc l'emploi d'un pattern de proposition ou de "phrase".

3°) Comme le plan, respectivement chaque modèle du plan, peut être réalisé dans 21 morphopatterns différents, le plan ou le modèle de plan n'équivalent pas à la proposition ou à la "phrase" d'attribution.

Cette constatation apparemment évidente signifie une différence fondamentale par rapport à toutes entreprises de traduction partant de la seule morphosphère.

Transformation, traduction, explication, verbalisation, réduction.

B. Généralités concernant la méthode de la traduction.

Les méthodes de la transformation, de l'explication et de la

[+] Nomosphère/morphosphère: d'après Glinz. Modèles appartenant à la nomosphère/patterns appartenant à la morphosphère: traduction française purement conventionnelle (entre nous et aujourd'hui) des mots allemands Modelle/Muster; prenez, si vous voulez bien, l'opposition modèles/paradigmes.

réduction opèrent dans la morphosphère ou la nomosphère, ou encore entre les deux sphères d'une même langue; s'agissant de traduction, d'une langue à l'autre, la méthode de la traduction s'ajoutera aux autres.

1°) Transformation .

a) Morphotransformations opérées dans le cadre de la morphosphère.
Il s'agit de transformations de formes dans le cadre de la morphostructure, des constituants formels étant utilisés.

b) Morphosémanticotransformations opérées dans le cadre de la morphosphère, compte tenu de la classe de facteurs sémantiques appelés par nous "modes de pensée" ("Denkweisen"); il s'agit des modes de pensée constituants de propositions, emboîtants, relevants, cyclopiques, nominaux, actionnels et désignants.

c) Morpho-nomotransformations, nomo-morphotransformations = applications de la morphosphère dans la nomosphère et vice versa. (Dans le sens du programme LIMAS: analyse et synthèse, dans le sens de Chomsky: transpositions de la structure de surface vers la structure en profondeur, "grammaire transformationnelle".)

d) Nomo-nomotransformations
  1°) dans le cadre d'une seule langue
  = transformation des formules de facteurs constituées à partir des facteurs sémantiques; cette transformation se fait par échange, addition ou soustraction de certains facteurs d'une formule;
  2°) entre la langue de départ et la langue d'arrivée
  = une formule de facteurs ne valant pas de la même façon pour la langue d'arrivée, devra être transformée, par coordination des formules en question des deux métalangues.

2°) Traduction.
On fait passer, sans modification, une formule de facteurs

dans la métalangue de la langue d'arrivée.

3°) Explication.

C'est l'explication du facteur sémantique d'une proposition pour lequel il n'y a pas de morphème dans la proposition.

Le manque de morphème correspondant peut être dû à:

1°) une ellipse,
2°) des relations contextuelles implicites,
3°) des cohérences sous-entendues existant entre différents facteurs sémantiques (frame of reference).

Il faut donc distinguer entre:

1°) l'explication d'ellipses,
2°) l'explication de relations contextuelles
3°) et l'explication de facteurs,

ces trois espèces pouvant se combiner.

4°) Verbalisation

= l'expression, en morphosphère, d'un facteur sémantique trouvé au moyen d'une explication, par un mot renfermant ce contenu.

5°) Réduction (ellipsation)

= le contraire de l'explication: un facteur figurant dans la formule de la métalangue ne sera ni verbalisé, ni exprimé par un morphème de la morphosphère. Tout en etant explicite dans la morphosphère, il ne sera pas biffé.

6°) Transformation explicitante.

Le facteur sémantique trouvé par explication transforme la formule factorielle: nomo-nomotransformation explicitante.
Il sera verbalisé ou formalisé: nomo-morphotransformation explicitante.

D. Application de la méthode à la proposition allemande d'attribution.

I. Différenciations formelles et sémantiques.

Dans l'acte de traduction, toutes les propositions d'attribution devront être transformées parce qu'en anglais, l'appartenance de la main à l'infirmière, et de l'épaule au malade, sont à exprimer. En même temps, il faudra transformer la forme dative[+] de la grandeur d'attribution. A cet effet, il faudra d'abord trouver, par explication, le facteur 'appartenance' contenu dans la proposition allemande d'attribution.

Ces propositions devront, en une première opération, être différenciées dans la morphosphère, en deux étapes et suivant les méthodes de la grammaire formelle conventionnelle; il apparaîtra vite que celles-ci ne suffiront pas à l'analyse du facteur en question.

1°) 1ère étape de la différenciation formelle
(morphopatterns) fig. I .

Les constituants formels servant à différencier sont les suivants: sujet, prédicat (= verbe), objet second, objet premier, "phrase" prépositionnelle, attribut. Il en résultera six patterns formels, d'après le schéma qui va suivre.
Les 714 propositions d'attribution correspondant à l'ensemble des 34 modèles d'attribution pourront être insérées dans ces 6 patterns de la façon suivante (il ne sera donné que des exemples):

patterns[++]:	propositions d'attribution
1	- die Schwester half dem Patienten - die Hand zitterte dem Patienten
2	- die Schwester verband dem Patienten die Hand - die Schwester reichte dem Patienten die Hand - die Schwester gab dem Patienten einen Stoß

[+] L'allemand possède, tout comme le latin, un troisième cas, le datif.

patterns: propositions d'attribution (suite)

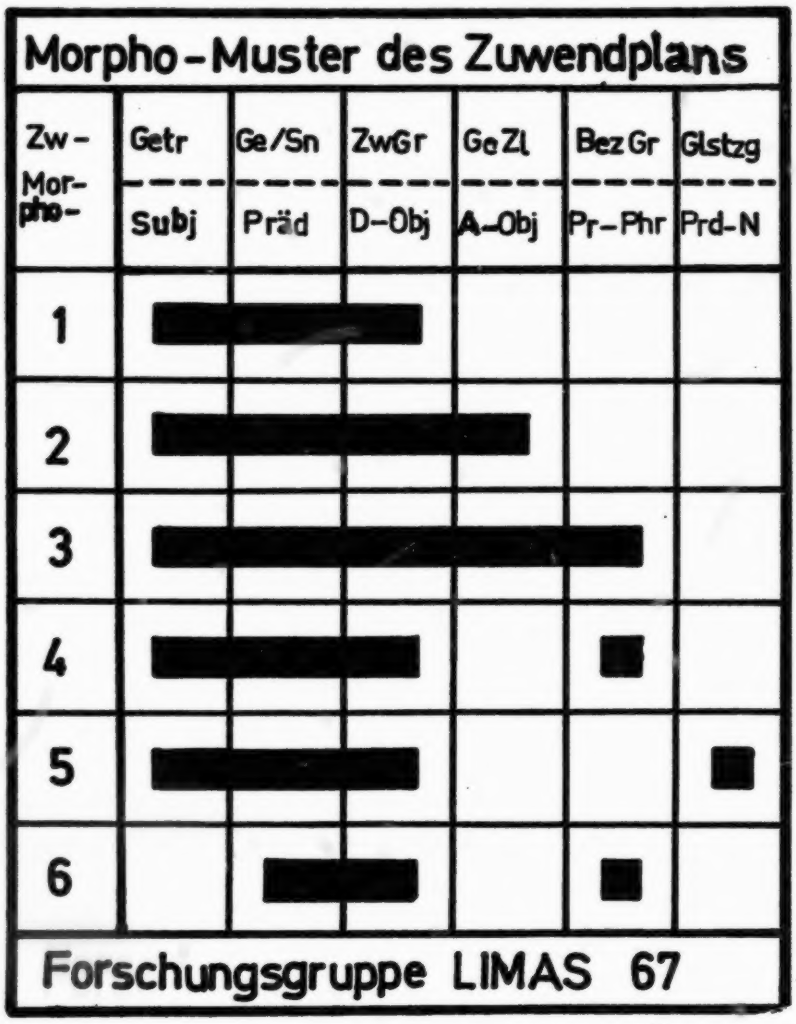
3 - die Schwester legte dem Patienten die Hand auf die Schulter
- die Schwester legte dem Patienten die Zeitung vor die Füße
- die Schwester reichte dem Patienten die Hand in den Wagen
- die Schwester schickte dem Patienten die Zeitung ins Haus
- die Schwester faltete dem Patienten die Hände auf der Brust

4 - die Schwester klopfte dem Patienten auf die Schulter
- die Schwester lag dem Patienten in den Armen
- die Schwester drohte dem Patienten mit dem Finger
- die Schwester drohte dem Patienten mit der Strafe
- die Schwester antwortete dem Patienten auf die Frage
- die Schwester sprach dem Patienten von dem Leben
- die Schwester klopfte dem Patienten mit der Hand auf die Schulter
- die Schwester schlug dem Patienten mit der Zeitung auf die Schulter

5 - die Schwester war dem Patienten behilflich
- die Schwester war dem Patienten eine Mutter

6 - der Schwester bangte um das Leben

Dans le pattern n° 4, on n'a pas tenu compte du nombre de "phrases" prépositionnelles. Tous les 6 patterns ne permettent pas de constater l'appartenance d'un contenu de membre à un autre si l'on part des constituants formels; en outre, les exemples établis à l'appui de ces 6 patterns contiennent des relations d'appartenances différentes.



Morpho-Muster des Zuwendplans

Zw-Morpho-	Getr	Ge/Sn	ZwGr	GeZl	BezGr	Glstzg
	Subj	Präd	D-Obj	A-Obj	Pr-Phr	Prd-N
1	■	■	■			
2	■	■	■	■		
3	■	■	■	■	■	
4	■	■	■		■	
5	■	■	■			■
6		■	■		■	

Forschungsgruppe LIMAS 67

2^{o}) 2ème étape de différenciation formelle

(morphosémanticopatterns) fig. 2 .

En différenciant les 6 morphopatterns (MP) d'après le nombre et la nature des "phrases" prépositionnelles, l'on se rendra compte qu'il faudra déjà faire appel à des considérations sémantiques, telles que espace-direction, espace-situation, instrument et objectif de relation (objet prépositionnel). Tout cela restera dans le cadre de la grammaire traditionnelle.

Nous appellerons ces différenciations morphosémanticopatterns (MSP), qui cependant ne permettront toujours pas l'établissement d'une analyse de la relation d'appartenance des éléments sémantiques de la proposition. Les 6 MP donneront 11 MSP.

MP n^{o} 3 bifurquera en MSP 3 et 4, MSP 3 comportant sujet, prédicat, objet second, objet premier, "phrase" prépositionnelle espace-direction et permettant, p.ex., les propositions suivantes:

- a) Die Schwester legte dem Patienten die Zeitung vor die Füße.
--b) Die Schwester reichte dem Patienten die Zeitung in den Wagen.

Dans a) il y a appartenance des pieds au malade; dans b) il n'y a pas d'appartenance.

MSP 4 présente la construction formelle de MSP 3, sémantiquement parlant il en diffère cependant par la "phrase" prépositionnelle espace-situation. Proposition possible:
- Die Schwester faltete dem Patienten die Hände auf der Brust.
Appartenances: de mains à malade, de poitrine à malade.
Aucune de ces appartenances n'est reconnaissable par des moyens morphologiques, ni même par les contenus adjoints aux constituants formels et qui sont le sujet agissant, l'activité, la grandeur d'attribution, l'objectif de l'activité et l'espace-situation (dans MSP 3 l'espace-direction).

Morpho - Inhalt - Muster des Zuwendplans

Morpho	Morpho - Inhalt	Subj	Präd	D-Obj	A-Obj	Präp - Phrase				Prd-N
		Getr	Ge/Sn	ZwGr	GeZl	Ort/R	Ort/L	Mittel	BzZl	Glstzg
1	1	■	■	■						
2	2	■	■	■	■					
3	3	■	■	■	■	■				
	4	■	■	■	■		■			
4	5	■	■	■		■				
	6	■	■	■		■		■		
	7	■	■	■			■			
	8	■	■	■				■		
	9	■	■	■					■	
5	10	■	■	■						■
6	11		■	■		■				

Forschungsgruppe LIMAS 67

Les mêmes considérations valent pour MSP 6:

- a) Die Schwester schlug dem Patienten mit der Hand auf die Schulter.
  Appartenance: épaule à malade, main à infirmière;
- b) Die Schwester schlug dem Patienten mit der Zeitung auf die Schulter.
  Appartenance: épaule à malade.

MSP 9:

- a) Die Schwester antwortete dem Patienten auf die Frage.
  Appartenance: question à malade;
- --b) Die Schwester sprach dem Patienten von dem Leben.
  Appartenance: aucune.

3°) Différenciation sémantique: modèles sémantiques du plan d'attribution. Fig. 3 .

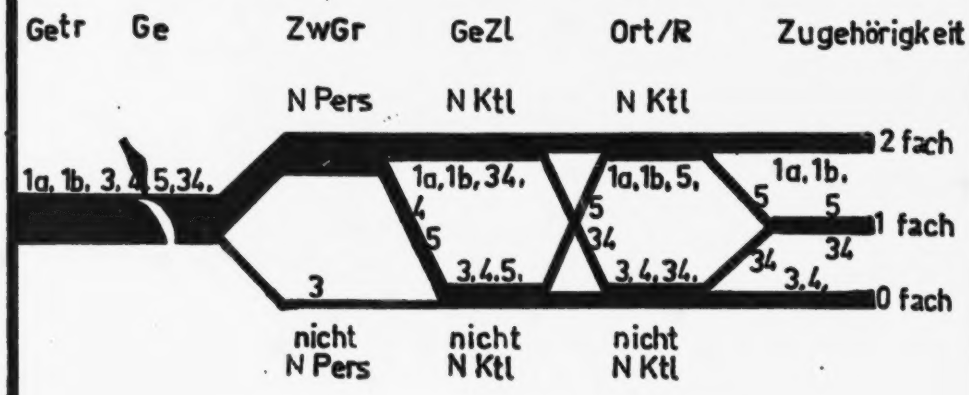
Ce n'est qu'en tenant compte des classes sémantiques nominales figurant en tant que membres de propositions, ainsi que de la rection des verbes, que l'explication de la relation d'appartenance des éléments sémantiques sera possible. Soit l'exemple de l'explication de MSP 3. Il suffira de retenir les classes sémantiques nominales personne/non-personne, et partie du corps/non-partie du corps; celles-ci seront rapportées aux noms dans lesquels apparaîtront la grandeur d'attribution, l'objectif de l'activité et l'espace-direction. On retiendra également les rections des verbes figurant dans nos exemples:

legen (poser), kämmen (peigner), geben (donner),
schicken (envoyer), strecken (tendre, donner);

ces rections serviront d'exemples des patterns de rection qu'elles représentent. De la sorte, le seul MSP 3 fournira 6 modèles sémantiques de plan d'attribution, soit:

1a - Die Schwester legte dem Patienten die Hand auf die Brust.
1b - Die Schwester kämmte dem Patienten das Haar aus der Stirne.

Explikation der Zugehörigkeit
Morpho-Inhalt-Plan Nr. 3


Getr
Ge
ZwGr
GeZl
Ort/R
Zugehörigkeit
N Pers
N Ktl
N Ktl
1a, 1b, 3, 4, 5, 34.
1a, 1b, 34,
4
5
3
3, 4, 5,
5
34
1a, 1b, 5,
3, 4, 34,
5
34
1a, 1b,
5
34
3, 4,
2 fach
1 fach
0 fach
nicht
N Pers
nicht
N Ktl
nicht
N Ktl

Modell 1a: **die Schwester legte dem Patienten die Hand auf die Brust**

Forschungsgruppe LIMAS 67

3 - Die Schwester gab dem Wagen einen Stoß gegen die Türe.
4 - Die Schwester schickte dem Patienten die Zeitung ins Haus.
5 - Die Schwester legte dem Patienten die Zeitung vor die *Füße*.
34 - Die Schwester streckte dem Patienten die *Hand* in den Wagen.

Les mots qui entrent dans une relation d'appartenance sont soulignés.

Il suit de ce qui précède que, selon la classe sémantique nominale figurant dans une case vide déterminée du plan, la relation d'appartenance des éléments du plan entre eux pourra être déterminée sans équivoque.
On pourra établir en particulier:

1°) s'il y a effectivement appartenance implicite;
2°) entre quels éléments sémantiques elle existe;
3°) combien de relations d'appartenance il y a;
4°) quel élément est, en l'occurrence, possesseur et quel autre objet possédé.

L'organigramme qui va suivre se limitera aux points 1 et 3.
La matrice complète compte 42 têtes de colonnes (facteurs) et 34 têtes de lignes (modèles sémantiques de plan d'attribution), ce qui empêche de la projeter ici. Les impressions opérées par la calculatrice démontrent cependant que l'algorithme fonctionne correctement. Celui-ci permet à la machine de réaliser les explications à l'ensemble des 34 modèles de plan d'attribution, quel que soit le pattern syntaxique où ils apparaîssent (ceux-ci sont du nombre de 21 et vont être exposés).

II. Explication, transformation, verbalisation de la proposition d'attribution.

Sur la base des informations que je viens de donner la calculatrice sera à même de procéder à l'explication du facteur d'appartenance et déterminera en même temps le possesseur et l'objet possédé.

En construisant, à l'occasion de la morpho-nomotransformation (m-ntr) et sur le plan de la nomosphère, une formule des facteurs sémantiques d'une proposition - base de notre système - elle pourra ajouter aux facteurs respectifs du modèle de plan d'attribution l'information possesseur (poss$^{\text{eur}}$) respectivement objet possédé (poss$^{\text{é}}$), en tant que facteurs expliqués.

Une telle transformation de formule (= nomo-nomotransformation: n-ntr) aura, p.ex., l'aspect suivant (nous prenons le modèle syntaxique analysé par M. L. Weisgerber, professeur de linguistique à l'Université de Bonn, et qui est notre modèle de plan d'attribution n° 2):

1°) Formule factorielle du modèle 2 de plan d'attribution, construite par m-ntr, dans la proposition énonciative indépendante.

- Die Schwester klopfte dem Patienten auf die Schulter.
(Les facteurs secondaires, qui ne sont pas indispensables à la présentation, ont été laissés de côté.)

$$\overset{(+1)}{\frac{(\text{r dét} \wedge \text{gr}^{1})\ \text{su ag}}{\text{nol}}} \wedge \overset{(+2)}{\frac{\text{act}}{\text{acl}}} \wedge \overset{(+3)}{\frac{((\text{r dét} \wedge \text{gr}^{2})\ \text{cl nom A})\ \text{gr attr}}{\text{nol}}} \wedge$$

$$\overset{(+4)}{\frac{((\text{r dét} \wedge \text{gr}^{3})\ \text{cl nom 4o})\ \text{esp-dir}}{\text{nol}}}$$

(+1)= renvoi déterminé à la grandeur 1, sujet agissant, le tout étant conçu nominalement;
(+2)= activité conçue actionnellement;
(+3)= renvoi déterminé à la grandeur 2, appartenant à la classe sémantique de noms A, grandeur d'attribution, le tout étant conçu nominalement;
(+4)= renvoi déterminé à la grandeur 3, appartenant à la classe sémantique de noms 4o, indication spatiale-direction, le tout étant conçu nominalement.

Liste provisoire des abréviations.

r dét $\wedge$: renvoi déterminé à
gr : grandeur
act : activité
cl nom : classe sémantique de noms
cl verb : classe sémantique de verbes
su ag : sujet agissant
attr : attribution
esp-dir : indication spatiale-direction
obj act : objet d'activité
nol : le tout étant conçu nominalement
acl : le tout étant conçu actionnellement
poss^{eur} : possesseur
$\text{poss}^{\text{é}}$: (objet) possédé

2°) Explication de la formule du modèle 2 de plan d'attribution.

$$\frac{(\text{r dét} \wedge \text{gr}^{1})\ \text{su ag}}{\text{nol}} \wedge \frac{\text{act}}{\text{acl}} \wedge \frac{(((\text{r dét} \wedge \text{gr}^{2})\text{cl nom A})\text{attr})\text{poss}^{\text{eur}}}{\text{nol}} \wedge \frac{(((\text{r dét} \wedge \text{gr}^{3})\ \text{cl nom 4o})\ \text{esp-dir})\ \text{poss}^{\text{é}}}{\text{nol}}$$

3°) Nomo-nomotransformation de l'explication du modèle 2.

a) Modification n° 1: substitut (représentant) d'appartenance. Nous insérons 'sub app' dans la formule:

$$\frac{(\text{r dét} \wedge \text{gr}^{1})\text{su ag}}{\text{nol}} \wedge \frac{\text{act}}{\text{acl}} \wedge \frac{(((\text{r dét} \wedge \text{gr}^{2})\text{cl nom A})\text{attr})\text{poss}^{\text{eur}}}{\text{nol}} \wedge \frac{(((\text{r dét} \wedge \text{gr}^{3})\text{cl nom 4o})\text{esp-dir})\ \text{poss}^{\text{é}}}{\text{nol}} \wedge \text{sub app}$$

et la proposition sera:

- Die Schwester klopfte dem Patienten auf seine Schulter.

Le pronom possessif sera trouvé (généré) au moyen du programme pronominal réversible, la calculatrice extrayant le genre et le nombre du possesseur, ainsi que le genre, le nombre et le cas de l'objet possédé.

Cette nomo-nomotransformation constitue maintenant le point de départ d'une n-mtr, pour le moment encore dans la langue allemande.

b) Modification n° 2a: grandeur d'appartenance, gr app. s'insérant dans la formule:

$$\frac{(((\text{r dét} \wedge \text{gr}^{3})\text{cl nom 4o})\text{esp-dir})\text{poss}^{\text{é}}}{\text{nol}} \wedge \text{gr app} = \text{poss}^{\text{eur}}$$

La calculatrice trouvera le possesseur dans la formule du modèle 2 de plan d'attribution, dans la partie de la formule

$$\frac{\underline{(((r\ \text{dét} \wedge gr^{2})\ cl\ nom\ A)\ attr)\ poss^{eur}}}{nol} .$$

Sur la base du programme transformationnel (coordinateur), elle rayera le facteur attribution figurant dans la formule de départ. Au lieu de $poss^{eur}$ en tant que grandeur il y aura alors gr^2 (= le malade) cl nom A $poss^{eur}$:

$$\frac{\underline{((r\ \text{dét} \wedge gr^{2})\ cl\ nom\ A)\ poss^{eur}}}{nol} .$$

Le mode de pensée nominal ne permettant qu'une expression nominale du possesseur, seul un complément déterminatif au génitif sera possible; c.-à-d., la n-ntr realisée de la sorte permettra respectivement exigera alors une n-mtr. La proposition sera:
- Die Schwester klopfte auf die Schulter des Patienten.

c) Modification n° 2b: par une m-mtr on pourra, si l'on veut, transformer cette forme en: - des Patienten Schulter. Afin de simplifier, l'on ne va pas représenter ce procédé restant dans le cadre de la morphosphère.

d) Modification n° 3: verbalisation.
Soit dit en passant que l'explication du facteur d'appartenance pourra également être utilisée pour l'expression littérale de celui-ci, c.-à-d., un mot ayant ce facteur dans son contenu pourra être introduit dans la proposition, ce qui équivaudra en même temps à une autre modification réalisée à l'occasion de la formule 2 expliquée du modèle. La proposition sera:
- Die Schwester klopfte auf die Schulter, die dem Patienten gehört.

E. Traduction de la proposition d'attribution vers l'anglais.

Ces nomo-nomotransformations une fois opérées dans le cadre de la métalangue allemande, l'on pourra réaliser, dans le procès réversible, en allemand et moyennant les nomo-morphotransfor-

mations, la nouvelle forme d'expression et, par conséquent, un modèle syntaxique différent,
Cette conversion du texte de départ pourrait déjà s'appeler "traduction réalisée dans le cadre d'une seule langue". En l'occurrence, elle est possible en tant que variété mais n'est pas nécessaire.

La traduction.

L'anglais cependant exige la conversion. Cette exigence sera faite et satisfaite, comme on vient de le montrer, par une matrice de comparaison et de coordination fonctionnant dans un coordinateur. Les modifications 1, 2a ou 2b, trouvées dans la métalangue allemande au moyen des n-ntr^{s} de la formule, pourront alors, sans autre modification, être transférées dans la métalangue anglaise.

Nous appellerons traduction un tel transfert inchangé d'une forformule factorielle de la métalangue de la langue de départ dans la métalangue de la langue d'arrivée.
Or, si la formule est modifiable par la seule matrice de coordination, sans qu'il faille aucune explication, il s'agit d'une n-ntr de coordinateur.

La formule de la langue d'arrivée trouvée par une méthode ou une autre commandera alors les n-mtr^{s}, ainsi que leurs parcours, à travers le processus de formulation de cette langue, vers une forme d'expression conforme au sens.

La proposition traduite sera, après la modification 1:
- The nurse tapped the patient on his shoulder
ou mieux, après la modification 2b:
- The nurse tapped the patient's shoulder.

Puisque, à l'occasion de cette traduction, il faut renoncer au facteur sémantique, typiquement allemand, d'attribution, mais qu'il faut, d'autre part, exprimer grammaticalement le facteur d'appartenance et laisser au verbe anglais son régime - objet premier, la traduction ne sera pas absolument identique de sens, objectif que ne saurait atteindre, en l'occurrence, le traducteur humain même.

En résumé:

1°) un modèle syntaxique anglais différent sera substitué au modèle allemand;

2°) chacun des deux modèles contient un élément sémantique que l'autre ne contient pas (dans la morphosphère);

3°) on a rendu la traduction possible en trouvant en allemand, par explication, le facteur d'appartenance exprimé en anglais par un élément morphologique;

4°) l'algorithme de traduction n'a été rendu possible que par l'analyse sémantique et n'a pu être construit sur la structure formelle; il vaut pour toutes les propositions allemandes d'attribution;

5°) comme et l'algorithme de traduction, et les processus de formulation fonctionnant dans chacune des deux langues sont réversibles, l'on pourra, sur la base du même système, traduire également de l'anglais vers l'allemand.

F. Développement dans le champ grammatical.

Il a été dit (voir supra p.5 sub 3°)) que chacun des 34 modèles de plan d'attribution ne saurait être réalisé, dans la morphosphère, que dans un des 21 patterns de propositions ou de "phrases". L'auteur a, dès 1955[+], montré les façons d'utilisation méthodique, pour l'enseignement de l'allemand aux étrangers et pour les lycées, de l'explication, de la traduction et de la verbalisation. Il y avait appelé "développement dans le champ grammatical" d'un facteur les différentes possibilités de réalisation, dans la morphosphère (= n-mtr^{s}), d'un facteur ou complexe de contenu.

Chaque facteur sémantique a, selon la nature des facteurs qui s'y joignent en le modifiant, plusieurs formes d'expression grammaticales; le facteur d'appartenance, p.ex.:

[+] Inhalte und Ausdrucksformen der deutschen Sprache, Frankfurt, Diesterweg, 1955 seq.

1°) la proposition d'attribution;
2°) le mot composé;
3°) le complément au génitif;
4°) le pronom possessif;
5°) des substantifs, adjectifs, verbes ou adverbes déterminés.

Réunis et en tant qu'ils expriment ce facteur, ils en représentent le champ grammatical. Comme, par contre, chacun de ces caractères formels est susceptible d'exprimer aussi d'autres facteurs que celui-ci, il a, inversement, un champ de différents facteurs sémantiques.
Nous allons, à titre d'exemple, examiner le champ grammatical d'un modèle de plan d'attribution, en tant que celui-ci pourra figurer dans différents patterns de propositions. L'ensemble de ses patterns constituera donc le champ grammatical du modèle de plan d'attribution. Comme nous ne pouvons exposer l'ensemble des 21 patterns possibles, nous nous limiterons encore une fois et ne retiendrons que 2, la proposition énonciative indépendante et la "phrase" complément déterminatif ("Attributivphrase").

Du modèle du plan d'attribution à la proposition énonciative indépendante.

De l'un à l'autre, il faut une n-mtr; la réalisation de celle-ci nécessite des facteurs supplémentaires de commande établissant la liaison de la formule de métalangue (nomosphère) à la forme d'expression (morphosphère).

Nous avons appelé ce groupe de facteurs "modes de pensée" puisqu'ils caractérisent la manière linguistico-formelle de penser un contenu. Ils ne sont pas nécessairement combinés avec des facteurs sémantiques déterminés mais avec des formes déterminées. Ce sont des données linguistiques choisies par le locuteur au cours du processus de formulation.
En tant que données linuistiques, ils représentent le répertoire linguistique des potentialités d'expression d'un contenu ou complexe de contenus, entre lesquelles le sujet par-

lant choisira. L'objet choisi et le mode de choisir sont des faits psychologiques et ne nous intéressent pas, l'intérêt linguistique étant réservé au répertoire linguistique donné.

Les facteurs de commande des modes de pensée rentrent dans le cadre de la métalangue, donc de la nomosphère.
Comment se fait une telle n-mtr? Nous recourons à la formule du plan d'attribution:

su ag $\wedge$ (act V être) $\wedge$ gr attr (grandeur d'attribution).

Un modèle du plan d'attribution, le modèle 1a, est libellé de la façon suivante:

$$(\text{gr/cl nom }\overset{1}{\text{A}})\text{su ag} \wedge (\text{act V}\ \overset{2}{\text{être}}) \wedge \overset{3}{\text{gr}}\ \text{attr/cl nom A}$$
$$\wedge\ (\text{gr/cl nom }\overset{4}{\text{4o}})\text{obj act} \wedge (\text{gr/cl nom }\overset{5}{\text{4o}})\text{esp-dir}\ .$$

Nous avons numéroté de 1 à 5 les éléments du modèle.

Le premier groupe des facteurs de commande qu'on y adjoint déterminera la formation en mots des différents éléments. La formule sera:

$$1 = \frac{((\text{r dét} \wedge \text{gr}^{1})\ \text{cl nom A})\ \text{su ag}}{\text{nol}}$$

(= renvoi déterminé à grandeur 1 de la classe sémantique de noms A, sujet agissant, le tout étant conçu nominalement.)

$$2 = \frac{\text{act}^{1}/\ \text{cl verb 8}}{\text{acl}}$$

(= activité de la classe sémantique de verbes 8, le tout étant conçu actionnellement.)

$$3 = \frac{((\text{r dét} \wedge \text{gr}^{2})\ \text{cl nom A})\ \text{attr}}{\text{nol}}$$

(= renvoi déterminé à grandeur 2 de la classe sémantique de noms A, attribution, le tout étant conçu nominalement.)

$$4 = \frac{((\text{r dét} \wedge \text{gr}^{3})\ \text{cl nom 4o})\ \text{obj act}}{\text{nol}}$$

(= renvoi déterminé à grandeur 3 de la classe sémantique de noms 4o, objet d'activité, le tout étant conçu nominalement.)

$$5 = \frac{((\text{r dét} \wedge \text{gr}^{4})\ \text{cl nom 4o})\ \text{esp-dir}}{\text{nol}}$$

(= renvoi déterminé à grandeur 4 de la classe sémantique de noms 4o, indication spatiale-direction, le tout étant conçu nominalement.)

Après avoir rempli les vides, gr$^{1-4}$ et act^{1}, avec les mots suivants:
gr^{1} = Schwester (infirmière)
gr^{2} = Patient (malade)
gr^{3} = Hand (main)
gr^{4} = Schulter (épaule)
act^{1} = legen (poser)
on aura la séquence de mots suivante:
- die Schwester legen dem Patienten die Hand auf die Schulter (l'infirmière poser au malade la main sur l'épaule).

Cette séquence n'est pas encore une proposition. On peut, en partant d'elle, former toute une série de formes différentes de propositions, p.ex.:

1°) Die Schwester legte dem Patienten die Hand auf die Schulter.
2°) Als die Schwester dem Patienten die Hand auf die Schulter legte.
3°) Die Schwester, die dem Patienten die Hand auf die Schulter legte.
4°) Leg dem Patienten die Hand auf die Schulter!
5°) Legte die Schwester dem Patienten die Hand auf die Schulter?
6°) Wer legte dem Patienten die Hand auf die Schulter?
7°) Wem legte die Schwester die Hand auf die Schulter?
8°) Was legte die Schwester dem Patienten auf die Schulter?
9°) Worauf legte die Schwester die Hand? +
10°) Legte doch die Schwester dem Patienten die Hand auf die Schulter!
11°) Wenn die Schwester doch dem Patienten die Hand auf die Schulter legte!
12°) Wenn die Schwester dem Patienten die Hand auf die Schulter legte, (subjonctif).
13°) Legte die Schwester dem Patienten die Hand auf die Schulter, (subjonctif).

+ Sous cette forme, l'appartenance de l'épaule au malade devient particulièrement évidente.

14°) Wenn die Schwester dem Patienten die Hand auf die Schulter legt.

15°) Wenn die Schwester dem Patienten die Hand auf die Schulter legte, (indicatif).

16°) Legt die Schwester dem Patienten die Hand auf die Schulter.

17°) Legte die Schwester dem Patienten die Hand auf die Schulter.

18°) Die dem Patienten die Hand auf die Schulter legende Schwester.

De telles formes différentes de propositions seront génerées moyennant le mode de pensée constitutif de propositions, qui comprend en tout 21 patterns différents. En l'occurrence, le pattern soit la proposition énonciative indépendante (= prop én ind). En allemand, elle est caractérisée avant tout par un verbe personnel rangeant en deuxième position. La formule, chiffrée, sera:

$$\frac{1 \wedge 2 \wedge 3 \wedge 4 \wedge 5}{\text{prop én ind}} \quad ,$$

ce qui fournira un pattern morphologique de plan d'attribution. L'algorithme du mode de pensée constitutif de propositions, programme partiel du processus de formulation, permettra alors la seule proposition:

- Die Schwester legte dem Patienten die Hand auf die Schulter.

Nous sommes ainsi arrivés du modèle 1a de plan d'attribution à MSP 3, et également à MP 3.
Prenons la diapositive 2 et voyons les intitulés des facteurs (têtes de colonnes)! Nous savons maintenant qu'aux facteurs sémantiques figurant sur la ligne inférieure, il faudra ajouter les facteurs mentionnés des modes de pensée afin que les facteurs figurant sur la ligne supérieure puissent être formulés comme proposition. Selon le facteur ajouté à ce mode de pensée constitutif de proposition, un autre pattern du même modèle de plan d'attribution sera formé.
L'adjonction, aux éléments du plan, de l'un ou l'autre facteur constitutif de proposition, ainsi que des modes de pensée nominale ou actionnelle constitue une n-ntr, tandis que la formulation moyennant l'algorithme ainsi déterminé représente une n-mtr.

(Certains facteurs secondaires ont été laissés de côté.) De la sorte, le modèle de plan d'attribution pourra être transformé dans chaque pattern de propositions d'attribution couvert par son champ grammatical.

II. Du modèle de plan d'attribution à la "phrase" complément déterminatif[+].

Soit la même séquence de mots obtenue à partir du modèle 1a de plan d'attribution. La formule symbolisée sera:

$$\underset{\text{dés}}{\underline{1 \wedge 2 \wedge 3 \wedge 4 \wedge 5}} \quad .$$

Le mode de pensée désignant (dés) implique que l'élément événement, qui est le support de la proposition, ne devra pas être formulé actionnellement (c.-à-d. sous forme de verbe) mais de mode désignant (c.-à-d. sous forme de participe). Sera donc sujet de la désignation (su dés) le sujet agissant; aussi le participe, affecté des facteurs secondaires nombre, genre et cas, sera-t-il formulé en fonction conjointe avec le sujet de la désignation, et le sujet agissant ne sera plus nécessairement sujet de proposition.
L'élément 1 de la formule sera donc:

$$\underset{\text{nol}}{\underline{(((\text{r dét} \wedge \text{gr}^{1})\ \text{cl nom A})\ \text{su ag})\ \text{su dés}}}$$

et l'élément 2: $\dfrac{\text{act}^{1}/\ \text{cl verb 8}}{\text{dés}}$.

Puisque le sujet de la désignation figure comme grandeur + mot exprimant un renvoi déterminé et que ceux-ci constituent un complexe sémantique étroitement lié, le mode de pensée emboîtant (emb) fonctionnera à son tour: le renvoi déterminé et la grandeur envelopperont l'élément désignant et tous les élé-

---

[+] En allemand "Attributivphrase". Dans les grammaires de l'allemand publiées en France, on appelle ces groupes ayant pour centre un participe, "propositions qualificatives", p.ex. Fourquet dans sa Grammaire de l'allemand, paragraphe 231, quoiqu'il s'agisse de déterminations.

ments de plan d'attribution qui en dépendent (et ceux qui n'en dépendent pas), l'expression du renvoi déterminé devenant premier, et la grandeur, second membre du cadre (me cad). La formule de l'élément 1 sera donc complétée comme suit:

$$\frac{\left(\underline{\text{r dét}/\ 1^{\text{er}}\text{ me cad} \underset{\text{emb}}{\wedge} \text{gr}^{1}/2^{\text{d}}\text{ me cad}}\text{ su ag}\right)\text{ su dés}}{\text{nol}}\ .$$

Tout le reste ne changera pas à l'occasion de la n-mtr. La "phrase" sera:

- Die dem Patienten die Hand auf die Schulter legende Schwester.

III. Du modèle de plan d'attribution à la proposition relative.

La formule valable dans la métalangue allemande pour la méta-"phrase" décrite ne saurait passer par traduction dans la métalangue anglaise, l'anglais n'admettant pas de telles "phrases" emboîtantes. Il faudra former une proposition relative:

- Die Schwester, die dem Patienten die Hand auf die Schulter legte.

Comme le processus reproduit ici dans le sens de la synthèse est réversible, l'on pourra développer la formule du modèle d'attribution en allemand, en partant de la "phrase".

Si l'on dote l'élément 2 des facteurs désignant, actionnel, proposition énonciative dépendante, et l'élément 1 du facteur su dés (sujet de la désignation), l'on pourra construire la proposition relative en allemand et en anglais, en partant de la même formule. La variante de celle-ci sera:

$$\text{élément 1} = \frac{(((\text{r dét} \wedge \text{gr}^{1})\text{ cl nom A) su ag) su dés}}{\text{nol}}$$

$$\text{élément 2} = \frac{(\text{act}^{1}/\text{ cl verb 8)}}{\text{dés acl prop en dép}}\ .$$

L'algorithme du mode de pensée constitutif de proposition visé par la formule de l'élément 2 conduira alors, compte tenu d'autres facteurs, secondaires, à la formation de la proposition relative correctement rapportée.

G. **Résumé**.

On a montré qu'il était possible:

1°) de résoudre des ellipses (explication);

2°) d'analyser des relations contextuelles sémantiques et de les utiliser aux fins de la traduction automatique (n-ntr8);

3°) de réaliser la traduction en dépendance structurelle et en même temps indépendamment de la structure, pour la raison qu'on dispose d'une méthode différenciée de transformation, de verbalisation, d'analyse sémantique et de classification de mots;

4°) de réaliser, sur la base des formules de métalangue, des générations systématiques;

5°) de constater finalement au moyen de la calculatrice, comme nous l'avons déjà montré à l'appui d'autres recherches, si un mot est employé au sens direct ou au sens figuré, et d'entamer par là l'analyse des métaphores et des idiotismes, par la voie du système et des contrastes constatés par rapport au pattern.

De la sorte, il paraît possible en principe d'apporter à des problèmes de traduction automatique non résolus à ce jour et considérés souvent comme insolubles tout court, une solution théorique, mais certainement une solution pratique poussée.

30

ESSAI D'UNE THEORIE SEMANTIQUE APPLICABLE AU TRAITEMENT DE LANGAGE

(MODELE "SENS - TEXTE").

A. JOLKOVSKY - I. MEL'ČUK
Institut de Linguistique
MOSCOU

RESUME

1. Le problème central de la linguistique est conçu comme la construction d'un modèle fonctionnel (susceptible d'être réalisé sous forme d'un programme pour ordinateur) du langage, c'est-à-dire, d'un système qui transforme n'importe quel sens donné en textes correspondants ou vice-versa. Partant du fait fondamental qu'en parlant on ne fait qu'exprimer un contenu préconçu, le modèle ici présenté ne se veut pas un système génératif dans le sens strict du terme, mais plutôt un système translatif avec entrée et sortie : si l'on introduit à l'entrée une "portion" de sens on obtient à la sortie tous les textes (de la langue modelée) qui portent le sens donné (synthèse) ; fonctionnant inversement le modèle produira pour chaque texte la notation symbolique de son sens (analyse).

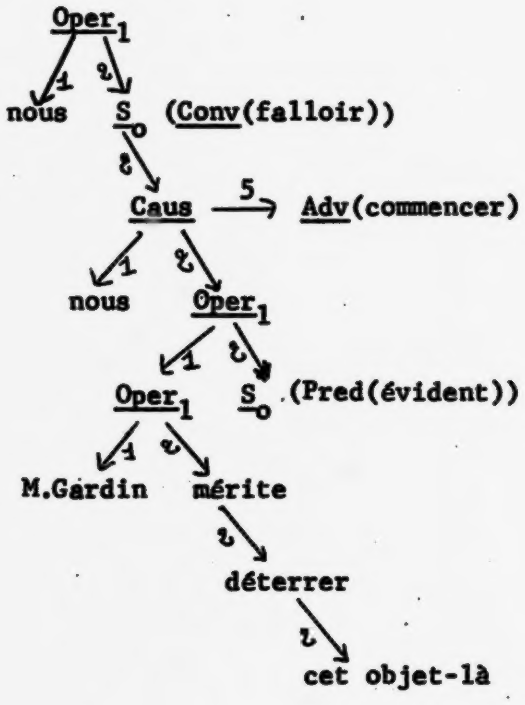
Le modèle "Sens - Texte" (dorénavant -MST) comporte trois niveaux principaux - morphologie, syntaxique et sémantique - dont seulement le dernier sera esquissé ici (sous l'aspect synthétique exclusivement).

2. Le composant sémantique du MST a pour but de transformer le sens (la notation sémantique donnée) en structures syntaxiques superficielles, c'est-à-dire, en ensembles de mots reliés par de liens syntaxiques, telles que "sujet-verbe fini" ou "préposition-nom régi" etc. Cette transformation s'opère en trois temps successifs : 1) de la "notation sémantique" (contenu d'un texte sous forme d'un graphe complexe constitué par des entités sémantiques élémentaires) vers les Basic-structures lexico-syntaxiques (BSLS, cf. 3) ; 2) d'une BSLS vers toutes les SLS synonymes (entre elles, aussi bien qu'avec cette BSLS), ou "structures profondes" ; 3) d'une SLS vers toutes les structures syntaxiques superficielles incarnant cette SLS. C'est de la deuxième opération que nous allons nous occupper ici, et notamment du mécanisme qui en est responsable et que nous

désigner par système de périphrases. On notera que sous périphrase nous entendons ici la transition d'une SLS aux SLS synonymes (et non d'une phrase aux phrases synonymes) une BSLS étant toujours une SLS).

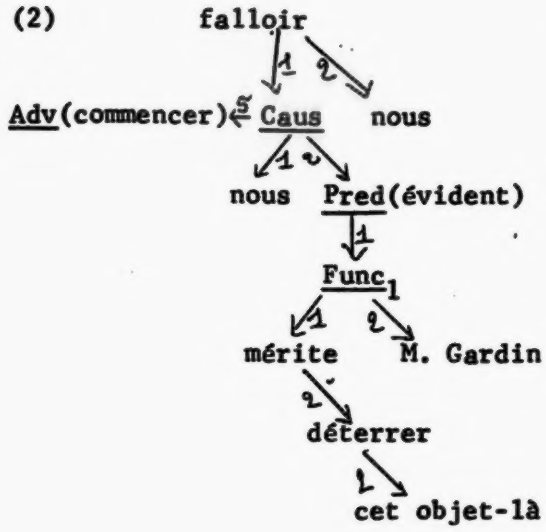
3. Une SLS (structure profonde d'une phrase) est une arborescence dont les branches dénotent des relations syntaxiques profondes et dont les noeuds sont des lexèmes profonds. La syntaxe profonde comporte six relations : 4 relations "actantielles" (cf. L. Tesnière), 1 relation attributive et 1 relation coordinative. Le lexique profond comprend des mots autonomes, ou mots clés, et des fonctions lexicales (cf. 4). Exemples de SLS synonymes (les fonctions lexicales sont imprimées en caractères gras) :

(1)

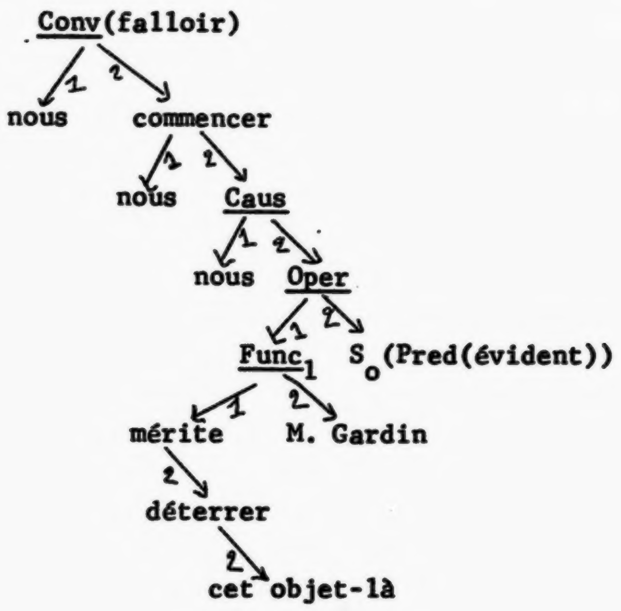

$\underline{Oper}_1$
1
2
nous
$\underline{S}_o$
(Conv(falloir))
2
Caus
5
Adv(commencer)
1
2
nous
$\underline{Oper}_1$
1
2
$\underline{Oper}_1$
$\underline{S}_o$
(Pred(évident))
1
2
M.Gardin
mérite
2
déterrer
2
cet objet-là

(Nous avons le devoir de mettre d'abord en évidence [le fait que] M. Gardin a le mérite d'avoir déterré cet objet-là.)

(2)


falloir
1
2
Adv(commencer)
5
Caus
nous
1
2
nous
Pred(évident)
1
Func1
1
2
mérite
M. Gardin
2
déterrer
2
cet objet-là

(3)


Conv(falloir)
1
2
nous
commencer
1
2
nous
Caus
1
2
nous
Oper
1
2
Func1
So(Pred(évident))
1
2
mérite
M. Gardin
2
déterrer
2
cet objet-là

(Il nous faut rendre d'abord évident [le fait que] le mérite d'avoir déterré cet objet-là revient à M. Gardin.)

(Nous devons commencer par mettre en évidence / le fait que / le mérite d'avoir déterré cet objet-là revient à M. Gardin.)

Dans l'ensemble des SLS synonymes, la SLS la plus simple et universelle (ces notions ne sont pas expliquées ici) est choisie pour représenter cet ensemble. Cette SLS est appelée Basic-SLS (sur son rôle, v. 7).

4. Une fonction lexicale est une dépendance sémantique entre un mot-clé (l'argument de la fonction) et un autre mot qu'on appellera le correlat lexical (la valeur de la fonction).
Ainsi :

Fonction	Argument	Valeur
Conv	précéder	suivre
(l'inversement des actants	falloir	devoir
Adv	commencer	d'abord
(l'adverbe dérivé)		
So	déterrer	déterrement
(le nom dérivé)	Pred (évident)	évidence
	tomber	chute
Oper$_1$	coup	porter
("faire", "être l'agent	accueil	réserver
de")	devoir, mérite	avoir
	S_o (Pred(évident))	être de [toute évidence]
	défaite	essuyer
Func$_1$	mérite	revenir à
("être l'action ou la	peur	posséder, gagner
propriété de")		
Pred	porteur	porter
("ètre + ...")		
Caus	liste	dresser
("créer", "faire avoir	symphonie	composer
lieu")	Func$_1$(peur)	faire à
	Pred(évident)	rendre
	Oper$_1$(S_o(Pred(évident)))	mettre en [évidence]
Magn	peur	grande, extrême, horrible, terrible
("très", "un grand degré	défaite	entière, complète, grande
de")		
	mérite	grand, éminent, supérieur
	besoin	grand, extrême, urgent, pressant

Jusqu'à présent une trentaine de fonctions lexicales ont été dégagées. Elles forment deux classes :

a) Les substituts - mots ou locutions susceptibles d'être employés dans le texte à la place du mot-clé, cette substitution n'exigeant que des modifications purement syntaxiques visant à sauvegarder le sens original (Conv, Adv, S_o dans la liste des exemples ci-dessus) ;

b) Les paramètres sémantiques - mots ou locutions susceptibles d'être employés avec le mot-clé pour exprimer d'une façon idiomatique certains sens ("très", "faire", "créer" etc. dans la liste ci-dessus).

Les valeurs de toutes les fonctions lexicales sont indiquées aux articles (mots-clés) respectifs d'un dictionnaire spécial ce qui permet une expression idiomatique des sens.

5. Les transformations réciproques des SLS sont effectuées par deux systèmes de règles :

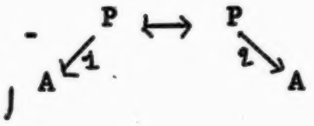
a) Règles lexicales (une cinquantaine) qui formulent des équivalences sémantiques en termes de fonctions lexicales. Ainsi, les transitions entre les SLS (1)-(3) sont fondées sur les règles suivantes :

	X	Y	Z		
1.	i_o(mot-clé)	$\leftrightarrow \underline{\text{Oper}}_1$	$\underline{S}_o(i_o)$	(1)	Nous accueillons... - Nous réservons un accueil à...
2.	i_o	$\leftrightarrow \underline{\text{Adv}}(i_o)$	-	(2)	Nous commençons par écrire ... - D'abord nous écrivons...
3.	i_o	$\leftrightarrow \underline{\text{Conv}}(i_o)$	-	(3)	Il nous faut lire... - Nous devons lire...
4.	$\underline{\text{Oper}}_1$	$\leftrightarrow \underline{\text{Func}}_1$	-	(3)	M. Gardin a le mérite... - Le mérite... revient à M. Gardin...

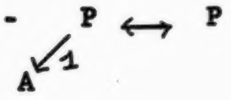
b) Règles syntaxiques (une vingtaine) qui ne sont en fait que les transformations possibles définies sur des arborescentes (ou sous-arborescentes) "profondes". Déclenchée par une règle lexicale à laquelle elle est rattachée

(v. les numéros dans la dernière colonne des règles lexicales), une règle syntaxique sert à remédier aux perturbations produites par les changements de vocabulaire dûs à la dite règle lexicale. Toute règle syntaxique est une combinaison de trois opérations-type primitives :

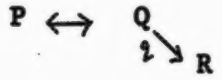
transplantation d'un noeud -



omission/addition d'un noeud -

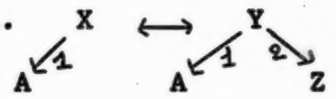


Fission/fusion des noeuds -

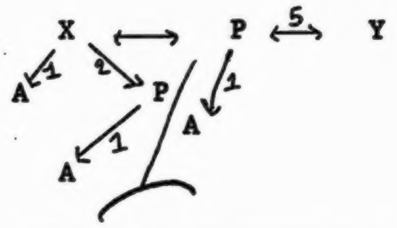


Les règles syntaxiques qui tiennent aux règles lexicales 1 - 4 sont :

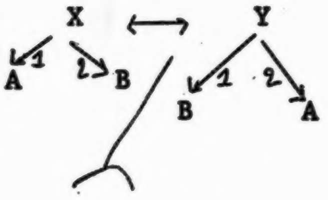
1.



2.



3.



6. Les règles mentionnées forment une espèce de générateur qui produit pour une SLS d'entrée le maximum des SLS synonymes. Ce générateur est couplé avec un sélecteur qui n'admet à la sortie que des SLS représentant des phrases tout à fait grammaticales. Le sélecteur contrôle les résultats de l'application des transformations synonymiques (formulées comme valables universellement) dans des cas concrets. Il consiste en un système assez ramifié de filtres - contraintes, exigeances, limitations de toute sorte etc. - disposés ça et là dans le système de périphrases et associés à des unités de dictionnaire, aux classes de mots etc. Les filtres visent l'absence d'une forme grammaticale d'un mot, le régime fort des verbes (tel complément est indispensable ou, au contraire, incompatible avec tel verbe), l'impossibilité d'une telle classe lexicale dans une telle construction etc. Les filtres sont absolument indépendants

des règles et excluent certains emplois d'une unité, quelle que soit la règle qui pourrait conduire à ces emplois.

7. Appliqué au traitement automatique de textes, le MST est en mesure de pourvoir à l'incongruence des structures des phrases des langues diverses, incongruence dont les recettes *ad hoc* ne sont qu'un remède de fortune. Le MST peut être incorporé dans un système, disons, de traduction automatique, de la façon suivante : la phrase de la langue-source doit être ramenée à sa SLS ; le système des périphrases transformera cette SLS dans la Basic-SLS correspondante ; celle-ci sera remplacée par la Basic-SLS (de la langue-cible) portant le même sens - ce qui doit être beaucoup plus facile que la traduction directe au niveau des phrases ; la BSL ainsi obtenue entrera, à son tour, dans le système des périphrases (de la langue - cible) qui va générer pour elle, une à une, les SLS synonymes jusqu'à ce qu'on aboutisse à une SLS acceptée par tous les filtres ; cette SLS sera développée en phrase par les mécanismes suivants du système. On peut espérer que le processus esquissé ici permettra d'éviter les discrépances interlinguistiques d'une manière naturelle et produire une traduction exacte, souple et assez idiomatique.

Le système des périphrases pour le russe, programmé par l'ordinateur "Ural-4" (par N. G. Arsentyeva et al.), est en train d'être expérimenté.

8. Quelques conclusions théoriques :

a) Le problème du sens est traité d'une façon opérationnelle, le sens n'étant dans le MST que l'invariante des transformations synonymiques.

b) L'opposition "profond - superficiel" est étendue au vocabulaire (fonctions lexicales *vs*. leurs valeurs verbales au niveau superficiel).

c) Les fonctions lexicales contribuent à la description systématique du vocabulaire révélant des proportions du type *protestation : vigoureuse* = défaite *défaite : complète* = *brouillard : épais* = *dormir : comme un plomb, à poings fermés* etc. ou *avoir :mérite* = *porter : coup* = *faire : pas* = *prêter, porter* : aide etc. (cf. /p/ : /b/ = /k/ : /g/ = /t/ : /d/...).

d) Un dictionnaire de type nouveau devient le pivot de la description linguistique. Chacun de ses articles doit contenir une liste de valeurs (30-50 environ) des fonctions lexicales pour le mot-clé. Les fonctions lexicales peuvent être considérées comme une sorte de questionnaire lexicographique universel.

e) La synonymie apparaît comme un phénomène beaucoup plus répandu qu'on ne le croit d'habitude, si l'on est prêt à la considérer à une échelle d'unités plus grande que celle des mots isolés. La synonymie n'est que l'interchangeabilité aux niveaux profonds du système. Quant à l'antinomie "équisignificance vs. interchangeabilité réelle dans un contexte concret", celle-ci est résolue par la stratification du modèle : des expressions résultant des substitutions sans limites à un niveau profond peuvent être exclues par des filtres d'un niveau superficiel.

AUTOLING : AN AUTOMATED LINGUISTIC FIELDWORKER

by Sheldon KLEIN
Computer Sciences Department
University of Wisconsin
MADISON-WISCONSIN 53706
U. S. A.

Sheldon Klein, Barbara G. Davis, William Fabens, Robert G. Herriot, Bill J. Katke, Michael A. Kuppin and Alicia E. Towster.

ABSTRACT

A system intended to act as a linguistic fieldworker via teletype interaction with a linguistically unsophisticated informant has been designed and is being programmed in extended ALGOL for the Burroughs 5500 and 8500 computers.

The system consists of the three major analytic components ; a program for performing morphological analyses and deriving a segmentation algorithm for sentences in any language ; a syntactic learning program that formulates context free and context sensitive phrase structure rules (monolingual learning component to be added later) ; and a machine translation program that learns to translate in both directions between the query language (English) and the language of the live informant via bi-lingual transformations.

The informant may be viewed as a fourth component, and is assumed to be able to read and write English in standard graphemics, and to be able to read and write his non-English language in a phonemic notation.

The analytic methods used by the AUTOLING are heuristic rather tha algorithmic, and hence do not guarantee complete success. The various components make use of protocols derived from the experience of live fieldworkers . The representations of these strategies are relatively separeted from the analytic mechanisms of the program and, accordingly, may be deleted, altered or incremented at the discretion of the system designers.

As a partial illustration of the system's operation, assume that the analystic process is at an intermediate stage, i.e. some morphological, syntactic and translation rules have been posited. Under the control of a monitor program, the system would test a nexly formulated syntactic rule by :

1- Generating (via usage of the new rule) a form implied by the grammar but never elicited from the live informant.

2- Translating the Hypothetical form into the query language (English).

3- Asking the informant via a teletype query to translate the English output of step 2.

If the informant's reply is equivalent to the form derived from step 2, the newly posited rule is maintened in the tenative grammar (although subject to later check). If the informant's reply deviates from the output of step 2, the new rule is treated as suspect and subjected to further verification procedures. If the deviating reply contains new morphological material, the monitor program instructs the morphological analytic component to update its analysis. The machine translation learning component would also recheck and update its rules.

The program as a whole continues its interaction with the informant indefinitly, using the just mentioned strategies as well as others. At any given time the system designers may interrupt the field work process and ask the machine to list its corpus and current rules.

Y. MARTEMIANOV et Y. MOUCHANOV
(Moscou, Laboratoire de Traduction
Mécanique de l'Institut des
Langues Etrangères)

Une notation sémantique et ses implications linguistiques

(Résumé)

1. Une notation est proposée en vue de fixer la structure sémantique d'un texte naturel de façon a en permettre le traitement automatique ultérieur, logique (déduction) ou linguistique (traduction). A la différence d'un langage documentaire (ou d'information), la notation proposée ne vise, en premier lieu qu'à éclaircir et ordonner la structure sémantique d'un énoncé naturel. Il s'agit, pratiquement, d'une forme de présentation, c'est-à-dire d'un 'moule" structuré, où est "coulé", après reconstructions, répartitions et regroupements nécessaires, le contenu d'un texte naturel.

2. Pour répondre aux buts poursuivis, la Notation est voulue :

a) dument *articulée* , c'est-à-dire disposant d'un nombre suffisant de catégories sémantiques générales, régulièrement réquisés par les textes naturels ;

b) suffisamment *exhaustive* - i.e. dégageant d'un énoncé naturel le maximum de données relatives aux catégories établies ;

c) rigouresement *réglementée* afin de révéler, à l'analyse, les interprétations fausses ou incomplètes.

d) susceptible de *différenciation* , i.e. permettant, pour un contenu donné, le passage d'une structuration globale et superficielle à une structuration plus profonde et détaillée ;

e) absolument *uniforme* , i.e. repartissant les données obtenues, à l'état global ou différencié, parmi une même série de catégories-"facettes".

3. Pour toute description élémentaire (DE, unité fondamentale de la notation) une même série de catégories est prévue, à savoir :

- *descripteurs* : prédicat à n termes (P^n_o), sujets (S_i) fixant la valeur des termes, et soi-disant "attributs présumés" (ou "présumés" tout court - (P^*_i)) qui, à la différence du prédicat, désignent les propriétés qu'un sujet doit impliquer pour devenir valeur d'un terme donné ; un présumé spécial (P^*_o) est rattaché au prédicat lui-même.

Ex : Un oiseau sort du nid =

$$= S_1 [\text{un oiseau}]\ S_2 [\text{le nid}]\ P_1^* [\text{se déplacer}]\ P_2^* [\text{espace clos}]\ P_0^2 [\text{sortir}]\ P_0^* [-]$$

A bird flies out of the nest =

$$= S_1 [\text{a bird}]\ S_2 [\text{the nest}]\ P_1^* [\text{fly}]\ P_2^* [\text{space closed}]\ P_0 [\text{get out}]\ P_0^* [\text{flying}]$$

- restricteurs (R) , désignant les limites temporelles (lim_1 - commencement , lim_2 - fin, lim_{-2} - suite), date et lieu de l'évènement décrit (α_0 , β_0 ...), à moins qu'ils ne fassent partie de l'évènement lui-même en tant que valeurs prévues par les termes du prédicat.

Ex. : les moineaux se faisaient rares dans les rues =

$$\left.\begin{array}{llll} = T & [\underline{\text{un temps passé}}] & R_{\text{date}} & (T, \alpha_0) \\ L & [\underline{\text{les rues}}]^{(\hat{t})} & R_{\text{lieu}} & (L, \alpha_0) \end{array}\right\} \alpha_0 = S_1 [\underline{\text{les moineaux}}]\ P_1^* [\underline{\text{élément}}]\ P_0 [\underline{\text{rare}}]$$

Les pierres grandissent en s'approchant =

$$= T [\text{on s'approche des pierres}]\ R_{\text{date}} (T, \alpha_0)\ \alpha_0 = S_1 [\overset{\text{les}}{\text{pierres}}]\ P_1^* [\text{objet}]\ P_0 [\text{grandir}]$$

- Indicateurs (I) - Indiquant le mode d'existance d'une description par rapport au sujet qui relate, imagine, désire ou suppose (etc...) l'évènement décrit.

Ex: : Qu'il entre :

$$= I [\text{vouloir } \alpha_0]\ \alpha_0 = S_1 [\text{il}]\ S_2 [\text{ici}]\ P_1^* [\text{se déplacer}]\ P_2^* [\text{espace clos}]\ P_0^2 [\text{entrer}]$$

Vous auriez... assassiné la République =

$$= I [\underline{\text{admettre } \alpha_0 \text{ pour un moment passé et le sachant non réalisé}}]$$

$$\alpha_0 = S_1 [\text{vous}]\ S_2 [\text{la République}]\ P_1^* [\text{personne}]\ P_2^* [\text{être animé}]\ P_0^2 [\text{assassiner}]$$

- Quantificateurs $(\forall, \exists, \exists^1)$ appliqués aux termes et restricteurs nondéterminés ;

- foncteurs $(\neg, \alpha, V_1 \rightarrow)$ servant à distribuer, parmi les descriptions élémentaires liées, les valeurs de "vrai" ou "faux".

Ex. : Autrefois les Français ne suivaient guère d'autres lois que celles du point d'honneur. = I $[\text{être presque sûr de } \alpha_\circ]$

$$\alpha_\circ = T[\exists\,\hat{t}] \quad R_{\text{date}} \quad (T, \beta_\circ)\, \beta_\circ = (S_2[\exists\ \text{LOIS}] \ S_1 \ [\text{les Français}] \ P_o^2\,[\text{suivre}]) \rightarrow (\text{lois} = \text{Lois du point d'honneur}).$$

L'implication et son dérivé -la causalité-($\rightarrowtail$) - permettent de réduire à leur valeur concrète les abréviations conventionnelles qui, dans le langage naturel, animent les notions abstraites : la conscience règne et ne gouverne pas

$$= S_1[\forall a] \ P_1^{*} \ [\text{personne}] \ (P_o \ [\text{conscient}] \rightarrow S_2[\forall b] \quad P_2^{*} \ [\text{société}] (P_0^2 \ [\text{régner}] \wedge \neg P_o^2[\text{gouverner}])).$$

"La curiosité naît de la jalousie"$= S_1[\forall a] \ P_2^{*}\underbrace{(\text{personne}]}\,(P_o\,[\text{jaloux}] \rightarrow P_o\,[\text{devenir curieux}]$

Remarque : Les notions de "prédicat", "termes", "quantificateurs" et "foncteurs" sont empruntées à la logique, sous réserve de la différence capitale qu'il y a entre leur emploi dans les sciences abstraites (étudiant les classes d'objets construites rationnellement) et dans les descriptions empiriques (objets réels disparates, classés intuitivement).

- Renvoyeurs (V_1 pour S_1, V_2^{*} pour S_2^{*} etc..) - renvoyant à un élément déterminé d'une description antérieure (V_2^{*} 1-) ou postérieure (V_2^{*} 1+)

Ex. : Vous avez changé de langage =

$$= \quad S_1\,[V_o^{*}\ 1+] \quad S_2 \quad [V_0^{*}\ 2+] \ P_o^2 \ [\text{différer}]$$

$$F_0^{*} \ : \ T[\exists\,\hat{t}] \ R_{\text{date}} \ (T, \alpha_\circ)\, \alpha_\circ = S_2\,[\forall \beta]\, S_1\,[\text{vous}] \quad P_1^{*}\,[\text{personne}] \ P_o^2\,[\text{parler}] \ P_o^{*}\,[\text{manière}]$$

$$F_0^{*} \ : \ T[\overset{\circ}{t}] \ R_{\text{date}} \ (T, \gamma_\circ)\, \gamma_\circ = S_2\,[\forall \delta]\, S_1\,[\text{vous}] \ P_1^{*}\,[\text{personne}] \ P_o^2\,[\text{parler}] \ P_o^{*}\,[\text{manière}]$$

c'est-à-dire "diffèrent la manière dont vous parliez autrefois et celle dont vous parlez maintenant (t).

- Refondeurs (F_1 pour S_2, F_2^{*} pour P_2^{*} etc..) - opérateurs présentant une description entière sous forme d'un seul de ses éléments, quitte à incorporer la description ainsi refondue dans une autre description (moyennant un renvoyeur).

Qui sait flatter, sait aussi calomnier =

$$= \quad F_1 : S_1[\exists a]\ S_2[\exists b]\ P_1^*[\text{personne}]\ P_2^*[\text{personne}]\ P_o^2[\text{savoir flatter}]$$

$$S_1[\forall_c,\ V_1\ 1-]\ S_2[\exists d]\ P_1^*[\text{personne}]\ P_2^*[\text{personne}]\ P_o^2[\text{savoir calomnier}].$$

(v. les transformations ultérieures à la page 6)

- opérateurs d'insistance : thème (⊥) - rhème (ᴚ), auxquels un sens logique précis est attaché, à savoir :

$$\perp a_o\,(P(a_o,\ b_o)) \underset{df}{\equiv} V_c\,(c = a_o \rightarrow P(c,\ b_o))$$

$$\text{ᴚ}\ b_o\,(P(a_o,\ b_o)) \underset{df}{\equiv} \exists_c\,(P(a_o, c) \propto c = b_o).$$

Ex. : l'oiseau, lui, vit dans un nid =

$$= S_1\,(S_1[\forall\,\text{oiseau}]\ S_2[\exists\ \text{nid}]\ P_1^*[\text{être animé}]\ P_2^*[\text{espace clos}]\ P_o^2[\text{vivre dans}]) =$$

$$\equiv P_1^*[\forall\,\text{être animé}]\ S_1[\forall\ \text{oiseau}]\ ([\text{être animé}] = [\text{oiseau}] \longrightarrow$$

$$\longrightarrow S_2[\exists\text{nid}]\ S_1[\text{être animé}]\ P_o^2[\text{vivre dans}]).$$

C'est dans un nid que l'oiseau vit =

$$= \text{ᴚ}\ S_2\,(S_1[\forall\text{oiseau}]\ S_2[\exists\ \text{nid}] \cdots P_2^*[\text{espace clos}]\ P_o^2[\text{vivre dans}]) \equiv$$

$$\equiv S_1[\forall\ \text{oiseau}]\ P_2^*[\exists\text{espace clos}]\ (S_1[\text{oiseau}]\ S_2[\text{espace clos}]\ P_o^2[\text{vivre dans}]\ \&\ [\text{espace clos}] = [\text{nid}]).$$

4. Foncteurs renvoyeurs et refondeurs constituent le minimum indispensable dont dispose la Notation pour marquer les relations syntaxiques entre les descriptions élémentaires, la syntaxe des catégories générales, inhérentes à la description, étant complètement déterminée par leur signification même.

Ex. : Nos grandes vertus finissent par nous lasser =

$$= \quad F_o \qquad S_1[\text{nous}]\ P_1^*[\text{personne}]$$

$$P_o[\text{vertueux}]$$

$$\rightarrowtail (1+,\ 2+) \qquad S_1[V_o\ \ 1-]\ -\ P_o[\text{excéder une norme}]$$

$$\forall T : T[\ V_oT\ 1+]\ R\ \lim_2(T,\alpha_o)\,\alpha_o \ = \ S_1\,[\text{nous}]\,P_1^*[\text{personne}]\ P_o\,[\text{vertueux}]$$

$$FT\ \ T\,[\overset{o}{t}]\,R\ \lim_1(T,\beta_o)\,\beta_o \ = \ S_1\,[\text{nous}]\,P_1^*\,[\text{être animé}]\,P_o\,[\text{las}]$$

i.e. Nous sommes vertueux et cela d'une façon tellement excessive qu'un moment vient où nous sommes plus vertueux et c'est le moment où nous commençons à en être las.

5. La séquence de descriptions élémentaires :

- ordonnées en colonne, avec leurs données réparties uniformément selon les catégories-facettes ;
- leurs relations mises en évidence ;
- la distribution de foncteurs, refondeurs et opérateurs d'insistance permettant de distinguer les descriptions principales et secondaires de l'énoncé ;
- et l'ordre de descriptions dans la colonne symbolisant la marche de l'exposé.

semble, d'une part, assez similaire aux propositions logiques pour ouvrir un passage à la déduction et, d'autre part, assez conforme à la structure sémo-syntaxique de l'énoncé naturel (SSS) pour servir de forme-cible à l'analyse sémantique.

6. L'analyse d'un énoncé visant à la notation est assurée, en premier lieu, par la notation elle-même, qui :

- possède une structure suffisamment rigoureuse pour rejeter les résultats incorrects ;
- dispose d'une certaine synonymie qui facilite le passage d'un énoncé naturel à une structure éminemment logique par exemple :

a) $S_1\ [\forall a]\quad P_1^*\,[M]\quad P_o\ [N] \rightleftarrows S_1\,[\forall a\ ,\ a \in M]\quad P_o\quad [N]$;

b) $S_1\ [a,\ a \in \{F_1 : S_1\,[\exists\, b]\ S_2[\exists c]\quad P_1^*\,[M]\quad P_2^*\,[N]\,P_0\ [Q]\}\,] \rightleftarrows$
$\rightleftarrows S_1\ [a]\ S_2\,[\exists\, c]\quad P_1^*\quad [M]\quad P_2^*\quad [N]\quad P_0\ [Q]$;

c) $S_1\,[\forall a\ ,\ a \in \{F_1 : S_1\ [\exists\, b]\ P_1^*\ [M]\ P_0\quad [N]\}]\ P_0\quad [Q] \rightleftarrows$
$\rightleftarrows S_1\ [\forall a]\quad P_1^*\quad [M]\quad (P_0\quad [N] \rightarrow P_0\quad [Q]\ )$.

Ex. : La structure sémo-syntaxique :

$$F_1\ :\ S_1\ [\exists\, a]\quad S_2\,[\exists\, b]\quad P_1^*\ [\text{personne}]\quad P_2^*\,[\text{personne}]\ \overset{2}{P_0}\ [\text{savoir flatter}]$$

$$S_1\left[\forall d,\ d\in\left\{V_1\ 1-\right\}\right] S_2[\exists c]\quad P_1^*[\text{personne}]\ P_2^*[\text{personne}]$$
$$P_o^2\ [\text{savoir calomnier}]\ .$$

obtenue pour l'énoncé "Qui sait flatter, sait aussi calomnier" (voir page 4 , devient l'objet des transformations suivantes :

$$S_1\ \left[\forall d,\ d\in\left\{F_1 : S_1[\exists\ \text{personne}]\ \ S_2[\exists\ \text{personne}]\ P_0^2\ [\text{savoir flatter}]\right\}\right].$$
$$S_2\ [\exists\,\text{personne}]\ P_0^2\ [\text{savoir calomnier}] \equiv$$
$$\equiv S_1[\forall\,\text{personne}]\ \ (S_2[\exists\ \text{personne}]\ P_0^2[\text{savoir flatter}] \rightarrow S_2\ [\exists\,\text{personne}]$$
$$P_0^2\ [\text{savoir calomnier}]\ )$$

(i.e. "Si une personne sait flatter quelqu'un, il existe des personnes qu'elle saura calomnier".

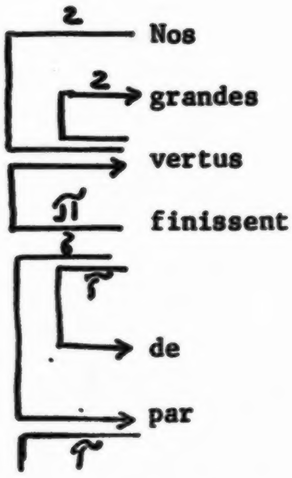
7. D'autre part, l'analyse s'appuie sur l'information sémantique rattachée à chaque mot et qui doit :

- être suffisamment articulée pour entrer, toutes les opérations nécessaires faites, dans une SSS correcte ;
- posséder, réunies dans une structure globale, toutes les structures actualisables, y compris les structures incompatibles.

Le souci principal de l'analyse sémantique d'un énoncé consiste, en gros, à éliminer, de l'information d'un mot pris dans son contexte syntaxique élucidé, les structures partielles

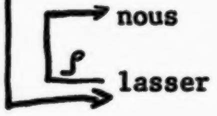
a) qui ne sont pas appuyées par les valences syntaxiques actualisées,
b) qui sont incompatibles avec d'autres structures, considérées comme plus convenables de par leur sens ;

Ex. : $(\pi, \rho\ \iota, \tau, \nu)$ symbolisant les valences syntaxiques, rsp . sujet, objet) direct, objet indirect, objet non-marqué, valence non-marquée ; sont surmontées d'un trait les structures partielles, éliminées de l'information sémantique du mot dans le contexte syntaxique donné : $\iota\downarrow$ – mot dépendant de ι , $\pi\uparrow$ – valence empruntée à un mot dominant) :



$$S_1\ [\text{nous}]\ \overline{S_2[\nu]}\,P_1^*\ [\text{personne}]\ P_o[\nu/\overline{\text{avoir}}]$$
$$S_1[\nu]\,P_o\ [\text{excéder une norme}]$$
$$S_1[\nu]\ P^*\ [\text{personne}]\ P_0\ [\text{vertueux}]$$
$$S_1[\pi]P_o\ [\exists d/\pi] \rightarrow T[\iota\downarrow]R_{\lim_2}(\ T, \alpha_o)$$
$$\alpha_o = S_1[\pi]\ P_o[\rho/\tau/\pi]$$

$$P_o[\nu]\ P_o^*[\tau]$$



$$S\,[\text{nous}]$$

$$\overline{S_1\,[\pi\uparrow]}\;P_o[\exists\alpha/\pi\uparrow]\to T\,[\mathring{t}]\,R_{\text{lim}_1}\,(T,\beta_\circ)\,\beta_\circ =$$

$$= S_1[\rho]\,P_1^{*}\,[\text{être animé}]\;P_o\,[\text{las}]$$

(v. la SSS résultante à la page 5 .)

8. Une attention particulière est accordée, dans l'analyse, aux informations sémo-syntaxiques auxilliaires qui, révélées chez un mot par le contexte, ne désignent qu'une opération à effectuer sur l'information sémantique de certains autres mots de l'énoncé ; Citons, pour exemples, les opérateurs INDUCT et ADDUCT :

α-β - INDUCT (γ) = 1) mettre l'information α du mot considéré dans les parenthèses, marquées de β, de l'information γ ;

2) supprimer le mot considéré (si, dans le contexte donné, il ne lui reste aucune information utilisable).

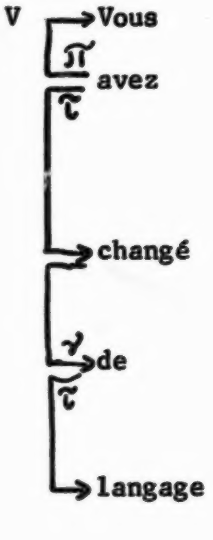
α-β - ADDUCT (γ) = 1) trouver, en remontant les mots dominant γ, la première information marquée de la valence α ;

2) combiner l'information trouvée avec celle du mot γ

3) mettre la combinaison obtenue dans les paranthèses marquées de α; de l'information β.

4) supprimer le mot considéré (si, ...)

Ex. : la SSS de l'énoncé "Vous avez changé de langage" (v. la page 3) est obtenue grâce à l'application des opérateurs en question constituant, dans le contexte donné, l'unique information des mots "vaez" et "de".



$$S\,[\text{vous}]\;P^{*}\,[\text{personne}]$$

$$S_1[\pi]\,S_2\,[\tilde{\tau}]\,P_2^{*}\;[\text{objet}]\;P_o\;[\overline{\text{avoir}}/\hat{\pi}-\hat{\pi}-\text{INDUCT}\;(\hat{\tau})]$$

$$S_1[\hat{\pi}]S_2\,[\pi]\,P_0^{2}\;[\text{différer}]$$

$$\begin{cases} T\,[\exists\,\hat{t}\,] & R_{\text{date}}\;(T,\alpha) \\ T\,[\exists\,\mathring{t}\,] & R_{\text{date}}\;(T,\alpha) \end{cases}$$

$$\overline{S_1[\nu]\,S_2\,[\hat{\tau}]\,P_1^{*}\,[\text{objet}]\,P_2^{*}\,[\text{lieu}]}$$

$$P_o\;[\,\overline{\text{de}}\,/\pi-\nu-\text{ADDUCT}\;(\tilde{\tau})]$$

$$F_o^{*}:\quad S_1\,[\exists\,a]\quad S_2\,[\,\exists\,b]\;\ldots$$

$$P_o^{2}\;[\text{parler}]\,P_o^{*}\,[\text{manière}]$$

9. L'appartenance à de diverses parties du discours des mots sémantiquement identiques, pas plus que le découpage syntaxique différent que subit une même information dans l'énoncé, ne sont pris en considération que pour autant que ces faits de langue permettent de desceller soit la subordination des descriptions élémentaires, soit la distribution d'opérateurs d'insistance (v. les exemples aux pages 3-5).

10. Le travail sur la Notation , entrepris pour orienter l'analyse automatique et permettre la déduction logique ultérieure, débouche vers la synthèse de textes naturels et de l'analyse documentaire, où des solutions, encourageantes se profilent.

33

SUMMARY OF SOME COMPUTATIONAL AIDS FOR OBTAINING A FORMAL SEMANTIC DESCRIPTION OF ENGLISH

by John Olney , Carter Revard, and Paul Ziff

The work reported herein was supported by contract F1962867C0004, Information Processing Techniques, with the Electronic Systems Division, Air force Systems Command, for the Advanced Research Projects Agency Information Processing Techniques Office.

System Development Corporation
2500 Colorado Avenue
Santa Monica , California 90406

Some Computational Aids for Obtaining a Formal Semantic Description of English

by

John Olney, Carter Revard, and Paul Ziff
System Development Corporation
2500 Colorado Avenue
Santa Monica, California 90406

The goal of a formal semantic description of a language may be characterized approximately as follows: to specify a set of lexical entries and a system of rules sufficient to provide appropriate readings for both individual sentences and multi-sentence connected discourses in the language, given appropriate structural descriptions of the sentences and discourses. (It cannot be required that only appropriate readings be derivable from the description, since semantic disambiguation often requires extralinguistic information. However, one would expect that the appropriateness of the readings derived will be correlated with their provision for inferences semantically warranted by the text segments in question.) As a contribution toward developing a formal semantic description of English (henceforth abbreviated as FSDE), we have transcribed two English dictionaries into machine-usable form and are developing programs for processing the information contained therein. The dictionaries are Webster's Seventh New Collegiate Dictionary (W7) and The New Merriam-Webster Pocket Dictionary (MPD), both published by G. & C. Merriam Co. Apart from pictorial illustrations and tables, the transcript of W7 is complete, even to the extent of preserving virtually all typographic distinctions made in the original.

We make the simplifying assumption that the word and phrase senses to be accounted for by the lexical entries and rules of an FSDE include all the senses described for entries in W7. Our routines for processing the W7 and MPD transcripts are designed to aid in the discovery of:

(a) rules for obtaining certain of the senses described for W7 entries from other senses described for the same entries or from senses described for other W7 entries from which the first were derived morphologically; and

(b) semantic components and rules for combining them to yield specifications of senses that cannot conveniently be obtained by the rules referred to in (a) above.

A first approximation to the set of lexical entries of an FSDE could be specified when the rules and semantic components referred to in (a) and (b) have been discovered; their discovery will of course require many, many man-years of effort and also close collaboration with researchers in English syntax. The basic contribution of our dictionary processing routines to this enterprise will be to permit semiautomatic generation of lexical fields for the main entries in W7. One type of lexical field which we plan to generate would display all words and phrases that are W7 entries and have been derived with the addition of a given affix or have been derived from a given base word by any morphological process. Appended to the field would be a printout listing the senses given in W7 for the words and phrases in the field. We have already constructed such derivational fields manually for a considerable number of affixes. By providing syntactic feature characterizations of the usages of the entries in each field, we have often been able to discern which element (if any) of the sense shift(s) accompanying the addition of the affix corresponds to a shift in syntactic function. In general, we anticipate that the derivational fields will prove quite useful as a computational aid for discovering the rules referred to in (a) above.

We also plan to generate semiautomatically a type of lexical field closely resembling what has traditionally been called a semantic field, viz., to construct for a given sense described at some entry in W7 a labeled network in which the nodes are other senses described in W7 that are closely related semantically to the given sense, and the labels indicate the type of relationship, e.g., synonymy, antonymy, etc.

A major objective of our current research is to determine, for the types of uses of the fields that we now foresee, exactly which relationships between word senses should be considered constitutive of semantic fields, and to what extent these relationships are signaled by machine-recognizable characteristics of W7 sense descriptions. Our results to date indicate that: (1) the semantic field output for a given word sense in the form of such network should be based on occurrences of the word in that sense, and occurrences of the nonfunction words used to describe that sense, in other sense descriptions provided in W7; and (2) a variety of other clues involving the syntax of defining phrases, stylistic regularities observed in W7, derivational morphology, the etymological information provided in W7, etc., should be taken into account before the field is output.

As part of our stylistic study of W7, we are working out conceptual analyses of the meanings of nonfunction words and phrases that are used very frequently in W7 definitions, starting with those chiefly used in defining derived senses. We will use these analyses to develop precise terminology for describing semantic relationships.

Probably the most frequent and serious errors in our mechanical tracing of fields will result from establishing links (in the network) between word senses whose descriptions in W7 contain a pair of word occurrences that are spelled the same, are functioning as the same part of speech, but are used in radically different senses. Clues based on stylistic regularities in W7 should help in disambiguating words occurring in sense descriptions, but the availability of MPD in machine-usable form should help still more. Although MPD describes only one-sixth as many senses as W7, nonetheless, for almost all word occurrences in W7 sense descriptions it defines the senses appropriate for those occurrences. Moreover, each MPD sense description is usually such a straightforward abbreviation of one or more W7 sense descriptions that our routines will generally be able to determine which W7 sense description(s) it abbreviates. Words occurring in W7 sense descriptions can then be disambiguated semantically to the extent of provisionally eliminating senses described for them in W7 which are not abbreviated in MPD. No doubt the tracing of semantic fields by our routines will still have to be monitored closely by the user, at least initially.

Edward Bendix has already shown in his recent dissertation how a plausible approximation to specifying semantic components of each of a group of words closely related semantically can be obtained by applying a series of semantic tests to all pairs in the group. Accordingly, we anticipate that our semiautomatically generated semantic fields will be used mainly to aid in discovering the semantic components and rules referred to in (b) above --more particularly for discovering components of words which, unlike kinship and color terms, do not belong to domains of the English vocabulary that are obviously well-structured.

As a further contribution toward discovering the elements of (b), we are attempting to arrive at a gross approximation to a set of semantic primitives for English by roughly the following procedure: Where A is the set of words occurring in W7 sense descriptions, find B, the set of words occurring in the sense descriptions of the members of A, and print out all pairs like 'infantile paralysis' and 'poliomyelitis', each of which occurs (as a crossreference) in the sense description of the other. Wherever possible eliminate one member of every such pair from B, then find C, the set of words occurring in the sense descriptions of the remainder of B, and iterate the process until the set most recently found is not significantly smaller than the set found immediately before it. The members of the smallest set found will then be scanned to see whether any further eliminations can be made. Preliminary results obtained by this procedure will be presented, together with the results of our stylistic analysis of W7 and our study of derivational processes. Applications of our routines for processing the W7 and MPD transcripts to lexicography, and to tasks in computational linguistics not directly concerned with semantics, will be discussed briefly.

Statistical and Linguistic Strategies in the Computer Grading of Essays

Ellis B. Page
University of Connecticut
Storrs, Conn., U.S.A.

Essay tests are used in the schools and colleges of all nations, and in major testing programs of national and even international size. Potentially, such essay tests are an important applied field for computational linguistics, and should eventually provide focus for much work. Yet in the past, little direct attention has been paid to such grading, although there are ways to begin investigation which would not necessarily require much linguistic knowledge beyond that now available.

Beginning in December of 1964, Project Essay Grade (PEG), at the University of Connecticut, has investigated the computer analysis and evaluation of student writing. In February, 1965, the project was given pilot funding by the College Entrance Examination Board of New York City, and in June, 1966, the United States Office of Education gave it much larger support. Through this period of preliminary investigation, certain problems have become much better understood (Daigon, 1966; Page, 1966, 1967). This paper discusses these problems, relates certain major findings to date, and outlines apparently promising avenues for future work by linguists, computer scientists, psychologists, and educators.

Background

It is useful to conceptualize the field of essay grading in two dimensions, as represented in Figure 1.

Figure 1

Possible Dimensions of Essay Grading

	I Content	II Style
A. Rating Simulation	I (A)	II (A)
B. Master Analysis	I (B)	II (B)

Any serious effort to grade essays must obviously face problems of "content" as in Column I, and of "style" as in Column II. Yet it is obvious that these columns are not mutually exclusive. Similarly, the rows are not mutually exclusive either, but their general meaning must be mastered to understand the work to date and the problems of the field. The first row refers to the simulation of the human judgment, without great concern about the way this judgment was produced. The second row refers to the accurate, deep, "true" analysis of the essay.

We have coined two terms to describe this difference. Since the top row is concerned with ap*prox*imation, we speak of the computer-variables employed as *proxes*. Since the bottom row is concerned with the true "*intrin*sic" variables of interest, we speak of such variables as *trins*. A *trin*, then, is a variable of intrinsic interest to the human judge, for example, "aptness of word choice". Usually a trin is not directly measurable by present computer strategies. And a *prox* is any variable measured by the computer, as an approximation (or correlate) of some trin, for example, proportion of uncommon words used by a student (where common words are discovered by a list look-up procedure in computer memory).

In the early part of our investigations, we concentrated on the right column and top row of Figure 1, looking for actuarial strategies, seeking out those proxes which would be of most immediate use in the simulation of the final human product, the ratings of stylistic factors.

For the first attempts, we evolved a general research design, which we have more or less followed to date:

(1) Samples of essays were judged by a number of independent experts. For our first trial 272 essays, written by students in Grades 8 to 12 in an American high school, and judged by at least four independent teachers. These judgments of overall quality formed the trins.

(2) Hypotheses were generated about the variables which might be associated with these judgments. If measurable by computer, and feasible to program within the logistics of the study, these computer variables became the proxes of the study.

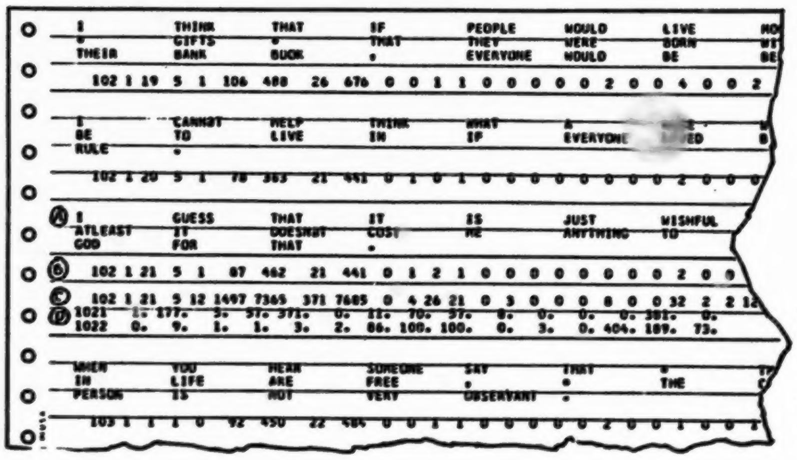
(3) Computer routines were written to measure these proxes in the essays. These were written in FORTRAN IV, for the IBM 7040 computer, and are highly modular and mnemonic programs, fairly well documented and available to computational linguists interested in using them or adapting them.

(4) Essays were prepared for computer input. In the present stage of data processing, this means that they were typed by clerical workers on an ordinary keypunch. They were punched into cards, and these cards served as input for the next stage.

(5) The essays were passed through the computer, under the control of the program which collected data about the proxes. The output was as appears in Figure 2.

Figure 2

PEG-IA Output



```
I           THINK       THAT        IF          PEOPLE      WOULD       LIVE        HO
*           GIFTS       *           THAT        THEY        WERE        BORN        WI
THEIR       BANK        BUCK        .           EVERYONE    WOULD       BE          BE
   102 1 19  5  1  106  488   26  676  0  0  1  1  0  0  0  0  0  2  0  0  4  0  0  2

I           CANNOT      HELP        THINK       WHAT        A           [illegible]   W
BE          TO          LIVE        IN          IF          EVERYONE    [illegible]ED B
RULE        .
   102 1 20  5  1   78  383   21  441  0  1  0  1  0  0  0  0  0  0  0  0  2  0  0  0

Ⓐ I         GUESS       THAT        IT          IS          JUST        WISHFUL
ATLEAST     IT          DOESNOT     COST        ME          ANYTHING    TO
GOD         FOR         THAT        .
Ⓑ  102 1 21  5  1   87  462   21  441  0  1  2  1  0  0  0  0  0  0  0  0  2  0  0
Ⓒ  102 1 21  5 12 1497 7365  371 7685  0  4 26 21  0  3  0  0  0  8  0  0 32  2  2 12
Ⓓ 1021   1. 177.   3.  57. 371.   0.  11.  70.  57.   8.   0.   0.   0. 381.   0.
  1022   0.   9.   1.   1.   3.   2.  86. 100. 100.   0.   3.   0. 404. 189.  73.

WHEN        YOU         HEAR        SOMEONE     SAY         THAT        *           TH
IN          LIFE        ARE         FREE        .           *           THE         C
PERSON      IS          NOT         VERY        OBSERVANT   .
   103 1  1  1  0   92  450   22  484  0  0  1  1  0  0  0  0  0  2  0  0  1  0  0  1
```

Figure 2 shows a piece of output from PEG-IA. Line A shows the way a sentence from the student essay is rewritten in 12-character double-precision computer "words" and stored in memory. Line B shows the summary of data for that sentence just analyzed. The first number is the essay identification. The other numbers of Line B are some counts from that sentence. Line C shows a summary of these counts, across sentences, for this whole essay. And Line D are these measures transformed in a number of simple ways, and ready for input into the final analysis.

(6) These scores were then analyzed for their multivariate relationship to the human ratings, were weighted appropriately, and were used to maximize the prediction of the expert human ratings. This was all done by use of standard multiple-regression programs.

The first analyses produced results as shown in Table 1. Here it is possible to read the list of proxes (Col. A), and their correlation, after transformation, with the human judgments of overall quality (Col. B). Col. C shows their contribution to the total multiple regression, and Col. D indicates the test-retest reliability of the proxes themselves, as discovered from two essays written by the same students, with about a month between writings.

Table 1*

Variables Used in Project Essay Grade I-A for a Criterion of Overall Quality

A. *Proxes*	B. *Corr. with Criterion*	C. *Beta wts.*	D. *Test-Ret. Rel. (Two essays)*
1. Title present	.04	.09	.05
2. Av. sentence length	.04	−.13	.63
3. Number of paragraphs	.06	−.11	.42
4. Subject-verb openings	−.16	−.01	.20
5. Length of essay in words	.32	.32	.55
6. Number of parentheses	.04	−.01	.21
7. Number of apostrophes	−.23	−.06	.42
8. Number of commas	.34	.09	.61
9. Number of periods	−.05	−.05	.57
10. Number of underlined words	.01	.00	.22
11. Number of dashes	.22	.10	.44
12. No. colons	.02	−.03	.29
13. No. semicolons	.08	.06	.32
14. No. quotation marks	.11	.04	.27
15. No. exclamation marks	−.05	.09	.20
16. No. question marks	−.14	.01	.29
17. No. prepositions	.25	.10	.27
18. No. connective words	.18	−.02	.24
19. No. spelling errors	−.21	−.13	.23
20. No. relative pronouns	.11	.11	.17
21. No. subordinating conjs.	−.12	.06	.18
22. No. common words on Dale	−.48	−.07	.65
23. No. sents. end punc. pres.	−.01	−.08	.14
24. No. declar. sents. type A	.12	.14	.34
25. No. declar. sents. type B	.02	.02	.09
26. No. hyphens	.18	.07	.20
27. No. slashes	−.07	−.02	−.02
28. Aver. word length in ltrs.	.51	.12	.62
29. Stan. dev. of word length	.53	.30	.61
30. Stan. dev. of sent. length	−.07	.03	.48

*Number of students judged was 272. Multiple R against human criterion (four judges) was .71 for both Essay C and Essay D (D data shown here). F-ratios for Multiple R were highly significant.

The overall accuracy of this beginning strategy was startling. The proxes achieved a multiple-correlation of .71 for the first set of essays analyzed and, by chance, achieved the identical coefficient for the second set. Furthermore, the beta weightings from one set of essays did well in predicting the human judgments for the second set of essays written by the same youngsters. All in all, the computer did a respectable, "human-expert" job in grading essays, as is visible in Table 2.

Table 2

Which One is the Computer?

Below is the intercorrelation matric generated by the cross-validation of PEG I

	Judges				
	A	B	C	D	E
A		51	51	44	57
B	51		53	56	61
C	51	53		48	49
D	44	56	48		59
E	57	61	49	59	

Here we see the results of a cross-validation. These are correlations between judgments of 138 essays done by five "judges," four of them human and one of them the computer. The computer judgments were the grades given by the regression weightings based on 138 other essays by other students. This cross-validation, then, is very conservative. Yet, from a practical point of view, the five judges are indistinguishable from one another.

However useful such an overall rating might be, we of course still wished greater detail in our analysis. We therefore broadened the analysis to five traits believed important in essays, adapted partly from those of Paul Diederich. They may be summarized as: ideas, organization, style, mechanics, and creativity. We had a particular interest in creativity, since some critics from the beginning have believed that the computer must founder on this kind of measure. "You might grade mechanics all right," someone will say, "but what about originality? What about the fellow who is really different? The machine can't handle him!"

Therefore, in 1966 we called together a group of 32 highly qualified English teachers from the schools of Connecticut to see how they would handle creativity and these other traits. Each of 256 essays was rated on a five-point scale on each of these five important traits, by eight such expert judges, each acting independently of any other judge. The teacher ratings were then analyzed, and it was found that the essay and the trait contributed significant variances, as did the trait-by-essay interaction. (perhaps the clearest demonstration of the ipsative profile). To investigate each of these five trait ratings, the same 30 proxes were again employed, with the results to be seen in Table 3.

Table 3

Computer Simulation of Human Judgments
For Five Essay Traits
(30 predictors, 256 cases)

A. Essay Traits	B. Hum.-Gp. Reliab.	C. Mult. R	D. Shrunk. Mult. R	E. Corr. (Atten.)
I. Ideas or Content	.75	.72	.68	.78
II. Organization	.75	.62	.55	.64
III. Style	.79	.73	.69	.77
IV. Mechanics	.85	.69	.64	.69
V. Creativity	.72	.71	.66	.78

Note:

Col. B represents the reliability of the human judgments of each trait, based upon the sum of eight independent ratings, August 1966.

Col. C represents the multiple-regression coefficients found in predicting the pooled human ratings with 30 independent proxes found in the essays by the computer program of PEG-IA.

Col. D presents these same coefficients, shrunken to eliminate capitalization on chance from the number of predictor variables (cf. McNemar, 1962, p. 184.)

Col. E presents these coefficients, both shrunken and corrected for the unreliability of the human groups (cf. McNemar, 1962, p. 153.)

In our rapidly growing knowledge, Table 3 may temporarily say the most to us about the computer analysis of important essay traits. Column A of course gives the titles of the five traits (more complete descriptions of the rating instructions may be supplied on request). Column B shows the rather low reliability of the group of eight human judges, computed by analysis of variance.

Here in Column B "creativity" is less reliably judged by these experts than are the other traits, even when eight judgments are pooled. And mechanics may be the most reliably graded of these five traits. Surely, then, humans seemed to have a harder time with creativity than with mechanics.

What of the computer? Column C shows the raw multiple correlations of the proxes with these rather unreliable group judgments. These were the coefficients produced by the standard regression program run by Dieter Paulus and myself. Column D simply shows the same coefficients after the necessary shrinking to avoid the capitalization on chance which is inherent with multiple predictors. Finally, in order for a fair comparison to be made among the traits, the criterion's unreliability should be taken into account, as in Column E. Here such difficult variables as creativity and organization no longer seem to suffer; the computer's difficulty is apparently in the criterion itself, and is therefore attributable to human limitations, rather than to machine or program limitations. Column E, then, exhibits what might be the expectable cross-validation from a similar set of essays, if predicting a perfectly reliable set of human judgments.

Current and Projected Problems

Of course, all this is a temporary reading taken in the middle of the research stream. Our investigators have also gone on with other strategies. Donald Marcotte (1967) has developed a phrase analyzer, and has discovered that clichés, as usually listed, were largely irrelevant to the judgment of such essays. Dieter Paulus (1967a) has studied the curvilinearity of proxes, and concluded that much elaborate statistical optimization may be a waste of time, and that the most major improvements should probably be made in other ways. He also has studied feedback to the student writer, using an on-line time-sharing console (Paulus, 1967b), as has also Michael Zieky. Another researcher, Jack H. Hiller (1967), has investigated quasi-psychological dimensions (including opinionation and vagueness) as predictors of the human judgments. Using techniques familiar from automatic content analysis (cf. Stone et al, 1966), he constructed lists of words and phrases to define the variables of psychological interest, and found these negatively correlated, as he predicted, with writing quality. And, in May, 1967, a sizeable improvement was made in the statistical accuracy, increasing the multiple-regression coefficient from about .71 to about .77, and improving the variance accounted for by around 20%. In other words, the newest programs apparently do better than the individual, expert English teacher.

The early strategies, then, have provided fertile ground for statistical investigation of essay grading, especially in the actuarial simulation of rating of style. But what of the deeper dimensions of stylistic analysis, and what of subject-matter content, as in essay questions in history, philosophy, or science?

Possible contributory linguistic strategies have been under more intensive study in recent months, with the advice and help of Susumu Kuno (1964), Stanley Petrick (Keyser and Petrick, 1967), John Olney (Olney and Londe, 1966; also see Harris, 1952) and others. (Of course these workers are not responsible for errors or misconceptions in the present paper.) Anticipated future strategies are currently summarized in Table 4. This table is based partly on work already accomplished in Project Essay Grade, partly on suggested minor adaptations of systems already working for others, and partly on projected programs which are not yet apparently operative in any system, but which do not seem impossibly difficult at the efficiency desired.

Table 4

Project Essay Grade

Hypothetical Complete Essay Grader

1. INPUT and PUNCH. Handwritten or typewritten or other raw response of the writer is converted for computer input.

2. SNTORG. Creates arrays of words and sentences as found in prose. This is just as performed in PEG-I.

3. DICT. Assignment of available syntactic roles to each word. This is currently done by many programs, but needs an expanded dictionary, and ambiguity resolver. At the same time, the semantic information will be stored in the workspace for reference of other parts of program. Availability of the tape-written Random House Dictionary (Unabridged) has been promised.

4. PARS. A modified Kuno (1964) program seems most promising, and is currently being programmed for both the 7094 and the 360 by workers at IBM. Alterations will be necessary to accept well-formed substrings.

5. REFER. This is intended to identify and encode the most likely referents of pronouns and other anaphoric expressions. (Cf. Olney and Londe, 1966). This process must employ both syntactic features and semantic information from DICT.

(Continued)

Table 4 (Continued)

---

6. KERNEL and STRUC. From the rewritten string output of (5), KERNEL would establish a set of elementary propositions, and STRUC would encode the relationships among these elements. This step would retain all the information of an essay in simplest possible units, yet would retain additional information about emphasis, subordination, causal relation, etc., among these units.

7. EQUIV. The elementary units would be augmented by the semantic information in DICT. To each word would be assigned a cluster of permissible synonyms, with weightings of semantic distance. This permits an analysis of redundance and emphasis in the essay, and permits a comparison of the content of the student essay with that of the key or master essay.

8. STYLE. Descriptions of the surface structure characteristics of the essay: parts of speech, organization of themes, types and varieties of sentence structure, grammatical depths, tightness of reference, etc; information about grammatical errors and strengths.

9. CONTNT. Comparison of the agreement of student and master essay, through measure of kernel hits and struc hits, these weighted by semantic distance of language chosen.

10. SCOR. Multivariate prediction of appropriate profile for the immediate purpose.

The limitations of space will permit only a few comments on this table, which may be seen as representing a hypothetical, ideal essay grader. For large grading systems, over established substantive content, it would be possible, for the key or master essay, to edit by hand the output from certain routines (especially REFER and STRUC). Of course, four of the most important routines listed in Table 4 are far from perfected in any existing programs. Ideally, they would assume better solutions to certain major, stubborn problems in computational linguistics.

Indeed, the steps in this hypothetical essay grader are close to the heart of the most persistent and troublesome problems in linguistics. Is it necessary that sentences be syntactically analyzed before mapping into deep structure? What is the proper role of semantics in such deep structure? How can the outside knowledge of the reader be incorporated into the machine analysis? (For some discussion of this problem, see Quillian, 1966). In general, how may we incorporate some of the intuitive richness which the literate human brings to his reading?

It is not expected that workers in essay grading will suddenly resolve all such questions. They may be recognized as those which so trouble linguists as to contribute to the recent official pessimism, in the United States, about the future of mechanical translation. After 15 years of effort, mechanical translation is still regarded as disappointing in quality, and virtually no sustained output of any machine program would be ordinarily mistaken for the work of a professional human translator.

On the other hand, the earliest attempts at essay grading by computer have, in a very limited way, leaped ahead of machine translation. And if the expert human ratings of high school essays may be regarded as an acceptable goal, then the machine program appears to have reached such a goal already. For that matter, improved performance, even superior to that of the individual human expert, appears to be immediately practicable as well.

The explanation of this advantage, of course, is that the problem of essay grading as attacked in the current work is much easier than the problem of machine translation. In translation, every nuance of the input string should be accounted for in the output string. In essay grading, only a certain portion of the input text needs to be accounted for, and the output does not depend on the existence of any large language-generating system. High quality machine translation apparently demands a fair portion of the total language-manipulating capability of the human, but essay grading may use only a fraction of it, and may process language in ways quite different from that of the human being. For example, our present programs have to date largely ignored order and sequence in the essays, although to the human the order of words is, of course, of crucial and unceasing importance.

Since essay grading can work with such fractional information, then, why pursue the deeper analysis of Table 4? Clearly, the purpose is not entirely the same as it would be for the usual linguist. At any discrete time in research, what is sought is not necessarily the perfect humanoid behavior, but rather those portions of that behavior which, given any current state of the art, will contribute optimally to efficient and practicable improvements in output. Indeed, regardless of the eventual perfection of deep linguistic behavior, for any specific application to essay grading, at any one moment, large portions of such available behavior may be irrelevant, just as it seems that ordinary human language processing does not usually call for our full linguistic effort.

Yet we regard it as eventually important to be able to perform these various kinds of advanced machine analysis when required. Therefore, the eventual uses of the ideal essay analyzer may require analytic capability as deep as may be imagined. Writing out suitable comments for the student, for example, will in some cases tax any system which may be foreseen.

Even approximate solutions to these problems, however, though unsatisfactory for certain scientific purposes, could make important contributions to the educational description and evaluation of essays. For such evaluation is itself probabilistic, limited by imperfect asymptotes of writer consistency and rater agreement. And such evaluation therefore does not require, to be practicable and satisfactory, the same deterministic perfection which has continued to elude and frustrate researchers in mechanical translation. There is a fundamental difference in goals, which must be realized. As has been demonstrated here, the output from much cruder statistical programs has already reached a quality not too remote from usefulness. The more advanced strategies currently seem, at least to the present workers, bright with promise.

REFERENCES

Daigon, Arthur. Computer Grading of English Composition. The English Journal, January, 1966, 46-52.

Harris, Z. S. Discourse analysis. Language, 1952, 8 (4), 474-493.

Hiller, Jack H., Page, E. B., and Marcotte, D. R. A Computer Search for Traits of Opinionation, Vagueness, and Specificity-Distinction in Student Essays. Paper read at the Annual Meeting of the American Psychological Association, Washington, D.C., September 2, 1967.

Keyser, S. J., and Petrick, S. R. Syntactic Analysis, 1966. (In press in a forthcoming book.)

Kuno, Susumu. Some characteristics of the Multiple-Path Syntactic Analyzer. Language Data Processing, Cambridge: Harvard Computation Laboratory, 1964. C6, 1-8.

Marcotte, Donald. The Computer Analysis of Cliché Behavior in Student Writing. Paper read at the Annual Meeting of the American Educational Research Association, New York, February 18, 1967.

McNemar, Quinn. Psychological Statistics, 3rd ed. New York: Wiley, 1962.

Olney, John and Londe, D. A research plan for investigating English discourse structure with particular attention to anaphoric relationships. Tech Memo mm-(L)-3256. Santa Monica, California: System Development Corporation. November 22, 1966. 17 p.

Page, Ellis B. The Imminence of Grading Essays by Computer. Phi Delta Kappan, January, 1966, 238-243.

Page, Ellis B. Grading Essays by Computer: Progress Report. Proceedings of the 1966 Invitational Conference on Testing Problems. Princeton, N.J.: Educational Testing Service, 1967. Pp. 87-100.

Paulus, Dieter. Problems of Nonlinearity in Grading Essays. Paper read at the Annual Meeting of the American Educational Research Association, New York, February 16, 1967a.

Paulus, Dieter. Feedback in Project Essay Grade. Paper read at the Annual Meeting of the American Psychological Association, Washington, D.C., September 2, 1967b.

Quillian, M. Ross. Semantic Memory. Cambridge, Mass.: Bolt Beranek and Newman, 1966.

References (Continued)

Stone, Philip J., Dunphey, Dexter C., Smith, Marshall S., and Ogilvie, Daniel M. The General Inquirer: A Computer Approach to Content Analysis. Cambridge: M.I.T. Press, 1966. Pp. 651.

Woods, William A. Semantics for a Question-Answering System. Paper read at the Annual Meeting of the Association for Machine Translation and Computational Linguistics. Atlantic City, N.J. April 21, 1967.

36

1967 INTERNATIONAL CONFERENCE
ON COMPUTATIONAL LINGUISTICS

An INTEGRATED LANGUAGE THEORY

S. Abraham

7; 13 Decembrie , Timisoara IV, Rumania
Rumanian Mathematical Society

ABSTRACT. The outlines of an INTEGRATED LANGUAGE THEORY are drawn, with grammar, semantics and text-theory as components. Problems of machine retrieval and machine translation are discussed on the basis of the outlined theory.

At present it is unanimously agreed that no computational linguistics can be developed without an (before - hand constructed) exact language theory. (The computer - handling of such an exact theory is in no way a transparent problem). The aim of the present paper is to outline such an exact integrated language theory.

The most promising basis for such a theory is the generative view on language. Unfortunately the generative theory of language is at present unsatisfactorily developed. First, this is true for generative grammars. But all other language theory constructions rely heavily on grammar (Katz & Fodor, Katz & Postal, Abraham & Kiefer).

At present generative grammar consists of two components : a phrase - structure component and a transformational one (I do not touch upon the problems of the phonemic and morphophonemic levels). In Chomsky's formulation the phrase - structure component necessarily contains context-restricted rules. But this has the unpleasant consequence that the most important decision problems are unsolvable (in this component). Besides this, no satisfactory solution is formulated for the generation of discontinous structures. Chomsky's last formulation of this component has even some more inconveniences. The transformational component has no exact formulation as the transformational rules are unsatisfactorily defined. The problem of analysis is also not satisfactorily solved.

For these reasons I propose a generative grammar also of two components : a matrix component and a transformational component. The matrix component consists of a regular elementary matrix grammar, containing only (ordered) elementary context - free rules , with solvable main decision problems, and which generates in a natural way discontinous structures. Within the transformational component the notion of transformational rule is exactly defined. The problem of an adequate analysis (of the generated sentences), the main problem of generative language theory, is satisfactorily solved by introducing the following analyses (graphs) : morphological (which usually is called the (phrase - structure) derivational graph), syntactical (not considered in Chomsky's variant), configurational (not considered in Chomsky's variant).

At present no (integrated) language theory can be conceived without a semantic component. In the theory which is outlined the semantic component is constructed as follows. A finite number of semantic categories (of the given language) are considered. Each word of the language is characterized by a proper semantic matrix and its definition . The dictionary of the language consists of triplets (word, proper semantic matrix, definition) called lexical issues. Each word of a configuration is characterized by its semantic matrix and its definition. On the basis of the semantic matrices the notion of semantic regularity of configurations is defined. The semantic regularity of a sentence is defined as the condition of the semantic regularity of all the configurations in the configurational analysis of the sentence. The sense of a(semantic regular) sentence is its configurational graph, with all the words substituted by their lexical issues. The notion of truth is introduced (in accordance with Tarski), and three types of analiticity are defined : grammatical, semantical and deductive analiticity.

On the basis of the above outlined theory an exact text component is developed, i.e. a component of the integrated language theory which deals with units larger than the sentence.

The outlined integrated language theory differs essentially from the constructions of Chomsky, Katz & Fodor, Katz et Postal, Abraham & Kiefer.

The exactly (and formally) constructed language theory permits to give a satisfactory solution to the theoretically based machine translation and machine retrieval. This last is formulated in the terms of A -oriented abstract (of a paper), ∅ - oriented abstract, N - oriented abstract, minimal N - oriented abstract, maximal N -oriented abstract and E - maximal superior N - oriented abstract.

BIBLIOGRAPHIE

1 - R. C. CROS, J. C. GARDIN, F. LEVY : "L'automatisation des recherches documentaires. Un modèle général : le SYNTOL." Gauthier-Villars, Paris, 1964, 260 p.

2 - A. BORILLO, M. COYAUD, J. VIRBEL : "Etude sur l'indexation automatique". Convention de recherche nº 65 - FR - 160 de la Délégation Générale à la Recherche Scientifique et Technique. Rapport semestriel nº 1, juin 1966, 37 p.

3 - M. COYAUD : "Une méthode sémantique d'indexation automatique de documents", International Conference on Computational Linguistics, New-York, mai 1965, 32 p.

4 - M. COYAUD et N. SIOT-DECAUVILLE, "L'analyse automatique des documents", Mouton, Paris 1967 (sous presse).

5 - "Un système général de documentation automatique, le "SYNTOL", Presses académiques européennes, Bruxelles, 1964, 4 tomes, 588 p.

37

LANGAGES "CONTEXT-SENSITIVES"
QUELQUES ASPECTS DE LEUR EXTENSION.

par J. FRIANT

SOMMAIRE.

Ce rapport rappelle les principales définitions concernant les grammaires de constituants, notamment les grammaires CS. Puis il introduit la notion de grammaires CS de reconnaissance, ce qui permet en particulier de démontrer simplement la stabilité de l'intersection relativement à la classe des langages CS. Plusieurs langages CS, de nature arithmétique, sont ensuite décrits; ainsi il est montré que l'ensemble des nombres premiers constitue un tel langage. En conclusion, l'énoncé de deux résultats permet d'inclure la classe des langages CF matricielles et celle des langages CFP dans la classe des langages CS.

§ 0. RAPPELS. NOTATIONS. [4]

0.1. Grammaires de constituants immédiats.

Ce sont des sytèmes formels dont le but est de décrire la construction des phrases de langages artificiels par modifications successives portant sur des suites de symboles. Formellement, une grammaire de constituants est la donnée d'un quadruplet:;

$$\mathscr{G} = (\mathscr{V}, \mathscr{V}_T, s, \mathscr{R})$$

- $\mathcal{V}$, appelé vocabulaire, est un ensemble fini de symboles;

- $\mathcal{V}_T$, vocabulaire terminal, est un sous-ensemble de $\mathcal{V}$, dont le complémentaire, $\mathcal{V}_N$, est appelé vocabulaire non-terminal;

- S est un symbole distingué de $\mathcal{V}_N$, désigné symbole initial.

Les symboles – éléments de $\mathcal{V}$ – seront représentés par des capitales latines habituellement indicées, et les suites finies de tels symboles, ou mots – éléments du monoïde libre $\mathcal{V}^*$ construit sur $\mathcal{V}$ – par des capitales latines surmontées d'u accent circonflexe.

A un couple ordonné de monts $(\hat{A},\hat{B})$ on associera le mot $\hat{C}$ obtenu en écrivant le mot $\hat{B}$ à droite du mot $\hat{A}$.

Formellement:

si $\hat{A} = A_1 \ldots A_i \ldots A_n$

$\hat{B} = B_1 \ldots B_j \ldots B_m$

alors $\hat{C} = \hat{A}\,\hat{B} = A_1 \ldots A_n\, B_1 \ldots B_m$.

On dira que $\hat{A}$, $\hat{B}$ et $\hat{C}$ sont des mots de longueur respectivement égale à n, m et n + m, et on écrira:

$$|\hat{A}| = n \quad , \quad |\hat{B}| = m \quad , \quad |\hat{C}| = n + m \; .$$

-$\mathcal{R}$ est un ensemble fini dont les éléments, appelés règles de production, sont des couples ordonnés de mots. Ainsi:

$r \in \mathcal{R}$: $r = (\hat{A},\hat{B})$ avec la restriction $\hat{A} \notin \mathcal{V}_T^*$.

0.2. Dérivation;

- On dira que $\hat{Y}$ dérive directement de $\hat{X}$, selon $\mathcal{G}$, par la règle de production $r = (\hat{A},\hat{B})$, s'il existe deux mots $\hat{G}$ et $\hat{D}$ tels que:

$$\hat{X} = \hat{G}\,\hat{A}\,\hat{D} \quad , \quad \hat{Y} = \hat{G}\,\hat{B}\,\hat{D} \quad .$$

On écrira : $\hat{X} \xrightarrow[\mathcal{G}]{\hat{G},\, r,\, \hat{D}} \hat{Y}$

ou simplement: $\hat{X} \xrightarrow[\mathcal{G}]{} \hat{Y}$

On dira que $\hat{Y}$ dérive de $\hat{X}$, selon $\mathcal{G}$, s'il existe une suite finie de mots $\hat{X}_0 , \hat{X}_1 , \dots , \hat{X}_n$ tels que :

$$\hat{X}_0 = \hat{X} \quad , \quad \hat{X}_n = \hat{Y}$$

et $\forall i,\ 1 \leq i \leq n \quad \exists\, r_i \in \mathcal{R} : \quad \hat{X}_{i-1} \xrightarrow[\mathcal{G}]{r_i} \hat{X}_i$

On écrira: $\hat{X} \xRightarrow[\mathcal{G}]{\hat{r}} \hat{Y}$ avec $\hat{r} = r_1 \dots r_i \dots r_n \in \mathcal{R}^*$

0.3. Langage.

Le langage engendré par la grammaire $\mathcal{G}$ est le sous-ensemble de $\mathcal{V}_T^*$ formé des mots dérivant de S, selon $\mathcal{G}$; on le notera $\mathcal{L}(\mathcal{G})$:

$$\mathcal{L}(\mathcal{G}) = \{ \hat{X} \mid \hat{X} \in \mathcal{V}_T^* ;\ \exists\, \hat{r} \in \mathcal{L}(\mathcal{G}) : S \xRightarrow[\mathcal{G}]{\hat{r}} \hat{X} \}$$

Deux grammaires seront dites équivalentes si elles engendrent le même langage.

0.4. Grammaires et langages context-sensitive.

Un sous-ensemble de $\mathcal{V}_T^*$ sera appelé langage context-sensitive s'il peut être engendré par une grammaire $\mathcal{G}$ dont toutes les règles sont telles que la longueur du premier membre est au plus égale à celle du second, i.e. :

$$\forall\ r \in \mathcal{R}, \quad r = (\hat{A},\hat{B}) : |\hat{A}| \leq |\hat{B}|$$

On remarquera que tous les langages context-free (CF) sont context-sensitive (CS).

On démontre $([7],[4])$ qu'un tel langage $\mathcal{L}$ peut toujours être engendré par une grammaire

$$\mathcal{G} = (\mathcal{V}, \mathcal{V}_T , S, \mathcal{R})$$

satisfaisant aux conditions suivantes:

. Outre le symbole initial S, $\mathcal{V}_N$ contient deux symboles particuliers S' et W; on note: $\mathcal{V}' = \mathcal{V}_N - \left\{ S \right\}$.

. Les règles se répartissent en trois groupes:

- les règles initiales: (S,SW) et (S,S')
- l'ensemble $\mathcal{R}'$ des règles binaires, de la forme: $(A_i\, A_j\, ,\, A_k\, A_l)$ aucun de ces symboles n'étant terminal,
- et l'ensemble $\mathcal{T}$ des règles terminales, de la forme:

$$(A,T) \ :\ A \in \mathcal{V}' \ ,\ T \in \mathcal{V}_T$$

Soit $\hat{X} \in \mathcal{V}_T^*$, $|\hat{X}| = n$; il résulte des conditions ci-dessus que $\hat{X}$ est une phrase de $\mathcal{L}$ si et seulement si on peut dériver $\hat{X}$ de $S'\, W^{n-1}$ à l'aide des règles de $\mathcal{R}'$ et de $\mathcal{T}$; plus précisément: $\hat{X} \in \mathcal{L}$ si et seulement si il existe $\hat{A} \in \mathcal{V}'^n$ tel que $\hat{A}$ dérive de $S'\, W^{n-1}$ par $\mathcal{R}'$ et $\hat{X}$ dérive de $\hat{A}$ par $\mathcal{T}$.

§ 1. GRAMMAIRE DE RECONNAISSANCE ASSOCIÉE À UN LANGAGE CS.

1.1. Soit $\mathcal{L}$ un langage CS engendré par une grammaire $\mathcal{G}$ du type rappelé ci-dessus. Nous allons définir une grammaire $\mathcal{G}_r$ équivalente à $\mathcal{G}$:

$$\mathcal{G}_r = (\mathcal{V}_r\ ,\ \mathcal{V}_T\ ,\ S\ ,\ \mathcal{R}_r)\ .$$

- Le symbole initial et les symboles terminaux sont ceux de $\mathcal{G}$. On trouve de plus les éléments de

$$\mathcal{V}'_r = \mathcal{V}' \times \mathcal{V}_T$$

représentés par: $\left[\frac{A}{T} \right]$ $A \in \mathcal{V}'$, $T \in \mathcal{V}_T$

- Les règles de $\mathcal{G}_r$ sont déduites de celles de $\mathcal{G}$:

. règles initiales:

$$\forall\, T \in \mathcal{V}_T \ :\ (S,\ S\left[\frac{W}{T}\right]) \ \text{et}\ (S,\ \left[\frac{S'}{T}\right])$$

. règles binaires, $\mathcal{R}'_r$:

$$\forall T_1 , T_2 \in \mathcal{V}_T : \left(\left[\frac{A_i}{T_1} \right] \left[\frac{A_j}{T_2} \right] , \left[\frac{A_k}{T_1} \right] \left[\frac{A_l}{T_2} \right] \right) \in \mathcal{R}'_r$$

si et seulement si $(A_i \, A_j \, , A_k \, A_l) \in \mathcal{R}'$

. règles terminales, $\mathcal{T}_r$:

$\forall T \in \mathcal{V}_T : \left(\left[\frac{A}{T} \right] , T \right) \in \mathcal{T}_r$ si et seulement si $(A,T) \in \mathcal{T}$.

Par extension si $\hat{A} = A_1 \dots A_i \dots A_n \in \mathcal{V}'^n$

et $\hat{X} = T_1 \dots T_i \dots T_n \in \mathcal{V}_T^n$

on écrira: $$\left[\frac{\hat{A}}{\hat{X}} \right] = \left[\frac{A_1}{T_1} \right] \dots \left[\frac{A_i}{T_i} \right] \dots \left[\frac{A_n}{T_n} \right]$$

Comme pour la grammaire $\mathcal{G}$, il résulte du choix des règles de $\mathcal{G}_r$ que $\hat{X} \in \mathcal{L}(\mathcal{G}_r)$ si et seulement si on peut dériver $\hat{X}$ de $\left[\frac{S' \, W^{n-1}}{\hat{X}} \right]$ à l'aide des règles de $\mathcal{R}'_r$ et de $\mathcal{T}_r$, plus précisément $\hat{X} \in \mathcal{L}(\mathcal{G}_r)$ si et seulement si il existe $\hat{A} \in \mathcal{V}'^n$ tel que $\left[\frac{\hat{A}}{\hat{X}} \right]$ dérive de $\left[\frac{S' \, W^{n-1}}{\hat{X}} \right]$ par $\mathcal{R}'_r$ et $\hat{X}$ dérive de $\left[\frac{\hat{A}}{\hat{X}} \right]$ par $\mathcal{T}_r$. Ainsi à toute S-dérivation de $\hat{X}$, selon $\mathcal{G}$, décomposable en (cf. ci-dessus) :

$$S \xRightarrow[\mathcal{G}]{} S' \, W^{n-1} \xrightarrow[\mathcal{G}]{(\mathcal{R}')} \hat{A} \xRightarrow[\mathcal{G}]{(\mathcal{T})} \hat{X}$$

correspond la S-dérivation de $\hat{X}$ selon $\mathcal{G}_r$:

$$S \xRightarrow[\mathcal{G}_r]{} \left[\frac{S' \, W^{n-1}}{\hat{X}} \right] \xRightarrow[\mathcal{G}_r]{(\mathcal{R}'_r)} \left[\frac{\hat{A}}{\hat{X}} \right] \xRightarrow[(\mathcal{G}_r)]{(\mathcal{T}_r)} \hat{X}$$

et réciproquement.

On a donc bien le résultat annoncé:

$$\mathcal{L}(\mathcal{G}_r) = \mathcal{L}(\mathcal{G}) = \mathcal{L} .$$

La grammaire $\mathcal{G}_r$ est dite de reconnaissance car si $\hat{X}$ est un mot terminal ($\hat{X} \in \mathcal{V}^*_T$, et $| \hat{X} | = n$) nous pouvons toujours envisager une S-dérivation, selon $\mathcal{G}_r$, telle que l'application des règles initiales aboutisse au mot $\left[\frac{S' W^{n-1}}{\hat{X}} \right]$; mais cette dérivation ne sera fructueuse que si $\hat{X}$ est une phrase du langage $\mathcal{L}$,. c'est-à-dire si la grammaire $\mathcal{G}_r$, par ses règles $\mathcal{R}'_r$ et $\mathcal{T}_r$, le "reconnaît" telle. On dira indifféremment que $\mathcal{G}$ engendre ou accepte $\hat{X}$.

N. B. : Comme dans le cas des grammaires CS ordinaires ([8],,[4]) on peut aussi envisager des grammaires CS de reconnaissance avec marquant d'extrémités de phrases, noté #. L'introduction d'un tel marquant ne modifie pas la capacité générative de nos grammaires.

1.2. Application: Intersection de langages CS:

En utilisant la notion de grammaire de reconnaissance nous allons prouver le résultat suivant ([8], [4]).

. L'intersection de deux langages CS est CS.

Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages CS sur le même vocabulaire terminal $\mathcal{V}_T$. On suppose donné pour $\mathcal{L}_i$ ($i = 1, 2$) une grammaire $\mathcal{G}_i$ du type indiqué en 0.4. ; on note S_i le symbole initial, S'_i, W_i les symboles spéciaux à $\mathcal{G}_i$. D'après le paragraphe 1.1. on peut définir une grammaire de reconnaissance $\mathcal{G}_{r_i}$ équivalente à $\mathcal{G}_i$ dont l'ensemble des symboles non-terminaux, mis à part S_i, est noté $\mathcal{V}'_{r_i}$ et l'ensemble des règles binaires (resp. terminales) $\mathcal{R}'_{r_i}$ (resp. $\mathcal{T}_{r_i}$).

On va construire une grammaire CS de reconnaissance, avec marquant

$$\mathcal{G} = (\mathcal{V}, \mathcal{V}_T, S, \#, \mathcal{R})$$

qui engendre $\mathcal{L}_1 \cap \mathcal{L}_2 = \{ \hat{x} \mid \hat{x} \in \mathcal{L}_1 \text{ et } \hat{x} \in \mathcal{L}_2 \}$.

- Vocabulaire: Le vocabulaire terminal de $\mathcal{G}$ est $\mathcal{V}_T$; le vocabulaire non-terminal $\mathcal{V}_N$ comporte en plus des symboles distingués S et #, les symboles de $\mathcal{V}'_{r_i}$ ($i = 1, 2$), i.e.:

$$\mathcal{V}_N = \{ S, \# \} \cup \mathcal{V}'_{r_1} \cup \mathcal{V}'_{r_2} .$$

- Principe du choix des règles de $\mathcal{G}$

Nous voulons définir une grammaire de reconnaissance qui n'accepte que les phrases appartenant simultanément à $\mathcal{L}_1$ et $\mathcal{L}_2$. Etant donné un mot terminal $\hat{X}$, nous vérifierons donc d'abord, à l'aide des règles de $\mathcal{R}'_{r_1}$, que $\hat{X}$ est une phrase de $\mathcal{L}_1$; si c'est le cas, et alors seulement, par des règles intermédiaires nous passerons à l'étude de la reconnaissance de $\hat{X}$ par la grammaire $\mathcal{G}_{r_2}$, à l'aide des règles de $\mathcal{R}'_{r_2}$. Et les règles de $\mathcal{C}_{r_2}$ nous permettront de passer au niveau terminal X, si et seulement si $\hat{X}$ appartient à l'intersection $\mathcal{L}_1 \cap \mathcal{L}_2$.

- Règles de $\mathcal{G}$:

. règles initiales

$$\forall\ T \in \mathcal{V}_T \ :\ \left(S, S\left[\frac{W_1}{T}\right] \text{ et } \left(S, \left[\frac{S_1}{T}\right]\right)\right)$$

. règles de "reconnaissance" de $\mathcal{G}_{r_1}$: $\mathcal{R}'_{r_1}$.

. règles intermédiaires

$\forall\ T, T' \in \mathcal{V}_T$.

$$\left.\begin{array}{l} \cdot\left(\left[\frac{A}{T}\right]\#, \left[\frac{W_2}{T}\right]\#\right) \\ \cdot\left(\left[\frac{A}{T}\right]\left[\frac{W_2}{T'}\right], \left[\frac{W_2}{T}\right]\left[\frac{W_2}{T'}\right]\right) \end{array}\right\} \begin{array}{l}\text{si et seulement si} \\ \left(\left[\frac{A}{T}\right], T\right) \in \mathcal{C}_{r_1}\end{array}$$

$$\cdot\left(\#\left[\frac{W_2}{T}\right], \#\left[\frac{S_2}{T}\right]\right)$$

. règles $\mathcal{R}'_{r_2}$ et $\mathcal{C}_{r_2}$ de la grammaire $\mathcal{G}_{r_2}$.

Le lecteur vérifiera facilement que le principe signalé plus haut se trouve respecté par le choix de ces règles. $\mathcal{G}$ accepte les phrases $\hat{X}$

appartenant simultanément à $\mathcal{L}_1$ et $\mathcal{L}_2$, et elles seules, ce qui démontre le résultat énoncé.

Nous allons présenter maintenant quelques exemples de langages CS.

§ 2. LANGAGES CS ARTIFICIELS.

Les langages étudiés dans ce paragraphe, sont définis sur le vocabulaire terminal $\mathcal{V}_T = \{ | \}$, réduit à un seul symbole. Une phrase, I^k, d'un tel langage, est entièrement caractérisée par sa longueur k, aussi, par abus de langage, dirons-nous parfois "le nombre k", pour le "mot I^k". On sait qu'un tel langage infini n'est CF que s'il contient un sous-ensemble "formant" une progression géométrique. J. P. Benzécri ([1], [2]) a généralisé la notion de langage CF en faisant intervenir des symboles et des mots à plusieurs insertions, et diverses opérations sur ces mots, en plus du produit de juxtaposition on parle de langage CFP. R. Guedj a montré [6] que tout langage CFP infini, sur le vocabulaire terminal $\{ | \}$, contient une progression arithmétique ou une progression de la forme: $\{ |^{w+uv^k} \mid k \in \mathbb{N} \}$. Ainsi le lecteur pourra vérifier facilement qu'aucun des langages étudiés dans ce paragraphe n'est CFP, a fortiori CF.

2.1. 1er exemple:

$$\mathcal{L}(i,j) = \{ |^{i^n + j^n} \mid ,n \in \mathbb{N} \}$$

i et j étant deux entiers distincts supérieurs à 1.

Soit $\mathcal{G}(i,j)$ la grammaire CS avec marquant

$$\mathcal{G}(i,j) = (\mathcal{V}(i,j), \mathcal{V}_T, S, \#, \mathcal{R}(i,j))$$

où: - $\mathcal{V}(i,j) = \{ S, \#, A_1, A_2, B_1, B_2, | \}$

- $\mathcal{R}(i,j)$ comporte :

. la règle initiale (S, A_1B_1)

. les règles de production

$(\# A_1 , \# A_2)$ (début)

$(A_2A_1 , A_1^i A_2)$; $(A_2B_1 , A_1^i B_2)$; $(B_2B_1 , B_1^j B_2)$

$(B_2 \# , B_1^j \#)$ (fin).

. et les règles terminales $(A_1 , |)$; $(B_1 , |)$.

Le lecteur vérifiera facilement que

$$\mathcal{L}(\mathcal{G}(i,j)) = \mathcal{L}(i,j)$$

Ce résultat serait évidemment susceptible de généralisation en considérant plus de deux indices (i,j).

2.2. $2^{\text{ème}}$ exemple:

$$\mathcal{L}(!) = \{ |^{n!} \mid n \in \mathbb{N} \}$$

Considérons la grammaire CS avec marquant

$$\mathcal{G}(!) = (\mathcal{V}(!) , \mathcal{V}_T , S , \# , \mathcal{R}(!)).$$

- $\mathcal{V}(!)$ comprend les deux symboles distingués S et #, les éléments de $\mathcal{V}_1$:

$$\mathcal{V}_1 = \{ A , A_1 , A_2 , B , C , C_1 , C_2 , D \}$$

et ceus de

$$\mathcal{V}_1' = \{ X' \mid X \in \mathcal{V}_1 \} , \quad \mathcal{V}_1'' = \{ X'' \mid X \in \mathcal{V}_1 \} , \quad \mathcal{V}_1''' = \{ X''' \mid X \in \mathcal{V}_1 \}$$

et le symbole terminal $|$.

Nous allons décrire les règles de la grammaire $\mathcal{G}(!)$ en indiquant les dérivations qu'elles permettent. L'idée directrice est la suivante: on passe de $|^{n!}$ à $|^{(n+1)!}$ en "copiant" (n + 1) fois le première phrase $|^{n!}$:

$$|^{(n+1)!} = \underbrace{|^{n!} \; |^{n!} \; \dots \; |^{n!}}_{(n+1)\text{ fois}}$$

Ainsi au niveau non terminal le passage de $|^{3!}$ à $|^{4!}$ correspondra à la dérivation:

$$\# AAAB^3 \# \xrightarrow[\mathcal{G}(!)]{} \# A^4 B^{20} \#$$

le nombre de symboles A permet de savoir à quelle étape on est rendu. On montrera comment les règles décrites permettent de réaliser la dérivation précédente.

I. Règle initiale (S, AA)

II. Règles permettant les dérivations du type:

$$\# A^3 B^3 \# \xrightarrow[\mathcal{G}(!)]{} \# DAAB^3 (C_1)^6 \#$$

. $(\# A , \# DA_1)$ -(début)

. (A_1A , AA_1C) ; (A_1B , BA_1C) : règles de "recopiage"

$$\left.\begin{array}{l} .\ (CA , AC) ; (CB , BC) \\ (C \# , C_1 \#) ; (CC_1 , C_1C_1) \end{array}\right\} \text{mise en place des nouveaux symboles}$$

$$\left.\begin{array}{l} .\ (A_1C_1 , C_1'C_1) \\ \forall\ X \in \mathcal{V}_1 , \forall\ Y \in \mathcal{V}_1 - \{ A_2 , D \} \\ (YX' , Y'X) \end{array}\right\} \text{passage à l'étape suivante}$$

Par la suite les symboles X et Y désigneront un symbole quelconque de $\mathcal{V}_1$, les restrictions faites seront seules indiquées.

III. Règles permettant les dérivations du type

$$\# DAAB^3(C_1)^6 \# \xrightarrow[\mathcal{G}(!)]{} \# DAA_2\, B^3\, (C_1C_2C_2)^6 \#$$

. (DA' , DA''_2) ou (A_2A' , AA''_2) : début de la nouvelle étape

. $(X''Y , XY'')$ $X \neq C_1$

. $(C_1''Y , C_1C_2Y'')$, $(C_1'' \# , C_1'C_2 \#)$: règles de "recopiage"

. $(C_2'' \# , C_2' \#)$

D'une façon générale s'il s'agit du passage de $|^{n!}$ à $|^{(n+1)!}$ cette étape recommence (n - 1) fois.

IV. Règles permettant les dérivations du type

$$\# DAA_2B^3 \; (C_1C_2C_2)^6 \; \# \xRightarrow[\mathcal{G}(1)]{} \# DA_3B^{20} \#$$

- $(A_2C_1' \, , \, A'''A''')$ ou $(A_2B' \, , \, A'''A''')$: un nouveau symbole A apparaît
  en vue d'une nouvelle étape éventuelle
- $(X'''Y \, , \, XY''')$ $X \neq C_1$ et $X \neq C_2$
- $(C_1'''Y \, , \, BY''')$
- $(C_2'''Y \, , \, BY''')$ ou $(C_2''' \# \, , \, B \#)$.

V. Enfin les règles

- $(XA''' \, , \, X'''A)$
- $(\# D''' \, , \, \# A)$

achèvent un développement complet (dans l'exemple choisi on aboutira à: $\# A^4B^{20} \#$), et l'on pourra recommencer un nouveau développemnt (on passerait à: $\# A^5B^{115} \#$) ou appliquer les :

VI. Règles terminales:

- $(S \, , \, |)$
- $(\# A \, , \, \# \, |)$ $(|A \, , \, | \, |)$
- $(|B \, , \, | \, |)$

On vérifiera que :

$$\mathcal{L}(\mathcal{G}(1)) = \mathcal{L}(1) = \{ |^{n!} \mid n \in \mathbb{N} \}.$$

2.3. 3$^{\text{ème}}$ exemple

$\forall i$ $i \geq 2$: $$\mathcal{L}^{(i)} = \{ |^{n^i} \mid n \in \mathbb{N} \} .$$

En s'inspirant de la grammaire $\mathcal{G}(1)$ on peut trouver une grammaire CS engendrant $\mathcal{L}^{(i)}$ [4]. Cela nous permet de donner dans le cadre des langages CS un énoncé du "théorème" de Fermat énonçant que les équations

$$x^n + y^n = z^n \quad ,$$

pour n entier supérieur ou égal à 3, n'admettent pas de solutions à valeurs entières. Il est facile de vérifier que cela revient à poser:

$$\forall\, i \,,\, i \in \mathbb{N},\, i \geq 3 \;:\; \left(\mathcal{L}^{(i)} . \mathcal{L}^{(i)}\right) \cap \mathcal{L}^{(i)} = \emptyset$$

Le produit $\left(\mathcal{L}_1 . \mathcal{L}_2 = \left\{ \hat{x} = \hat{x}_1 . \hat{x}_2 \mid \hat{x}_1 \in \mathcal{L}_1 \text{ et } \hat{x}_2 \in \mathcal{L}_2 \right\}\right)$ et l'intersection (cf. § 1.2.) étant stables relativement à la classe des langages CS, on peut définir une grammaire CS engendrant le langage du premier membre de l'égalité précédente. Malheureusement le problème général, de savoir si le langage engendré par une grammaire CS est vide, est indécidable.

Nous avons aussi démontré [5] que le langage

$$\mathcal{L}(p) = \left\{ |^{p} \mid p \in \mathbb{N},\, p \text{ premier} \right\}$$

est CS.

Mais ici nous allons présenter une grammaire CS engendrant les nombres premiers écrits sous forme décimale.

§ 3. NOMBRES PREMIERS DECIMAUX ET LANGAGES CS.

L'ensemble des nombres premiers, écrits sous forme décimale usuelle, est un langage CS.

DEMONSTRATION:

Considérons la grammaire CS avec marquant:

$$\mathcal{G} = \left\{ \mathcal{V},\, \mathcal{V}_T \,,\, S \,,\, \# \,,\, \mathcal{R} \right\}$$

où: - le vocabulaire terminal, $\mathcal{V}_T$, est l'ensemble des chiffres,

$$\mathcal{V}_T = \left\{ 0,\, 1,\, 2,\, \ldots,\, 8,\, 9,\, \right\}$$

- le vocabulaire non terminal, $\mathcal{V}_N$, comporte en plus des deux symboles distingués S et # , tous les sumboles du type: $\left[\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right]$ où

c_1 , c_2 , c_3 sont des chiffres arbitraires l'un quelconque d'entre eux pouvant éventuellemnt être indicé par «'» ou «"» , plus précisément:

$$\mathcal{V}_N = \left\{ S,\ \#,\ \left[\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right],\ \left[\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c'_1 \end{array}\right],\ \left[\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c''_1 \end{array}\right],\ \left[\begin{array}{c} c_3 \\ \hline c'_2 \\ \hline c_1 \end{array}\right],\ \left[\begin{array}{c} c_3 \\ \hline c''_2 \\ \hline c_1 \end{array}\right],\ \left[\begin{array}{c} c'_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right],\ \left[\begin{array}{c} c''_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right] \,\middle|\, c_1, c_2, c_3 \in \mathcal{V}_T \right\};$$

- les règles de la grammaire $\mathcal{G}$ vont être divisées en trois groupes.

I. Règles initiales:

$$.\ \# S \longrightarrow \# SS$$

$$.\ \# S \longrightarrow \# \left[\begin{array}{c} c \\ \hline 0 \\ \hline c \end{array}\right] S \qquad c \in \mathcal{V}_T - \{0\}\ .$$

$$.\ \left[\begin{array}{c} c_1 \\ \hline 0 \\ \hline c_1 \end{array}\right] SS \longrightarrow \left[\begin{array}{c} c_1 \\ \hline 0 \\ \hline c_1 \end{array}\right]\left[\begin{array}{c} c_2 \\ \hline 0 \\ \hline c_2 \end{array}\right] S$$

$$\left[\begin{array}{c} c_1 \\ \hline 0 \\ \hline c_1 \end{array}\right] S\# \longrightarrow \left[\begin{array}{c} c_1 \\ \hline 0 \\ \hline c_1 \end{array}\right]\left[\begin{array}{c} c_2 \\ \hline 2 \\ \hline c_2 \end{array}\right] \# \qquad c_1\ ,\ c_2 \in \mathcal{V}_T$$

$$.\ \# S \# \longrightarrow \# \left[\begin{array}{c} u \\ \hline 2 \\ \hline u \end{array}\right] \# \qquad u \in \mathcal{V}_T - \{0\ ,\ 1\}\ .$$

Faisons ici l'hypothèse que le symbole S n'intervient pas dans les autres règles et que ces dernières n'agissent que sur les symboles du type $\left[\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right]$ indicés ou placés à l'extrémité gauche: il en résulte que toute #S#-dérivation commencera par une séquence:

$$\#S\# \longrightarrow \#SS\# \longrightarrow \#S^{n-1}\# \longrightarrow \# \left[\begin{array}{c} c_n \\ \hline 0 \\ \hline c_n \end{array}\right] S^{n-1} \# \longrightarrow \ \ldots$$

$$\ldots \longrightarrow \# \left[\begin{array}{c} c_n \\ \hline 0 \\ \hline c_n \end{array}\right] \ldots \left[\begin{array}{c} c_p \\ \hline 0 \\ \hline c_p \end{array}\right] S^{n-p+1} \# \longrightarrow \ldots \longrightarrow \# \left[\begin{array}{c} c_n \\ \hline 0 \\ \hline c_n \end{array}\right] \ldots \left[\begin{array}{c} c_2 \\ \hline 0 \\ \hline c_2 \end{array}\right]\left[\begin{array}{c} c_1 \\ \hline 2 \\ \hline c_1 \end{array}\right] \#,\ c_n \neq 0$$

- ou simplement $\#S\# \longrightarrow \#\left[\begin{array}{c} u \\ \hline 2 \\ \hline u \end{array}\right]\#$ $u \in \mathcal{V}_T - \{0 , 1\}$

Ces dérivations faites, les règles initiales ne pourront plus être appliquées.

II. Règles de transformation:

Elles seront décrites formellement ci-dessous; signalons ici qu'il s'agit de règles conservant la longueur et que leur but est de déterminer si, oui ou non, le nombre N,

$$N = c_n c_{n-1} \cdots c_2 c_1$$

est premier. Dans ce but N va être "divisé" par les nombres D, compris entre 2 et N:

$$D = d_l d_{l-1} \cdots d_2 d_1$$

Ainsi, en cours de dérivation on passe par le mot non terminal:

$$\left[\begin{array}{c} c_n \\ \hline 0 \\ \hline c_n \end{array}\right] \cdots \left[\begin{array}{c} c_l + 1 \\ \hline 0 \\ \hline c_l + 1 \end{array}\right] . \left[\begin{array}{c} c_l \\ \hline d_l \\ \hline c_l \end{array}\right] \cdots \left[\begin{array}{c} c_2 \\ \hline d_2 \\ \hline c_2 \end{array}\right]\left[\begin{array}{c} c_1 \\ \hline d_1 \\ \hline c_1 \end{array}\right] ,$$

que l'on représenter schématiquement par $\left[\begin{array}{c} N \\ \hline D \\ \hline N \end{array}\right]$, $2 \leq D \leq N$.

- Si N est un multiple de D on obtient le mot $\left[\begin{array}{c} 0 \\ \hline D \\ \hline N \end{array}\right]$, i.e. :

$$\left[\begin{array}{c} 0 \\ \hline 0 \\ \hline c_n \end{array}\right] \cdots \left[\begin{array}{c} 0 \\ \hline d_l \\ \hline c_l \end{array}\right] \cdots \left[\begin{array}{c} 0 \\ \hline d_2 \\ \hline c_2 \end{array}\right]\left[\begin{array}{c} 0 \\ \hline d_1 \\ \hline c_1 \end{array}\right] ,$$

et l'on bloque la dérivation, N n'étant pas premier.

- Si N n'est pas divisible par D on passe à la "division" par (D + 1), c'est-à-dire au mot non terminal $\left[\begin{array}{c} N \\ \hline D + 1 \\ \hline N \end{array}\right]$

Il en résulte que si et seulemnt si N est premier on aboutit au mot non terminal $\left[\begin{array}{c} N \\ \hline N \\ \hline N \end{array}\right]$, i.e. :

$$\left[\begin{array}{c} c_n \\ \hline c_n \\ \hline c_n \end{array}\right] \left[\begin{array}{c} c_{n-1} \\ \hline c_{n-1} \\ \hline c_{n-1} \end{array}\right] \cdots \left[\begin{array}{c} c_2 \\ \hline c_2 \\ \hline c_2 \end{array}\right] \left[\begin{array}{c} c_1 \\ \hline c_1 \\ \hline c_1 \end{array}\right]$$

dont on dérivera le nombre premier N par les règles terminales.

III. Règles terminales. c, c_1, c_2 sont des chiffres quelconques:

$$\left[\begin{array}{c} c \\ \hline c \\ \hline c \end{array}\right] \# \longrightarrow c \, \#$$

$$\left[\begin{array}{c} c_1 \\ \hline c_1 \\ \hline c_1 \end{array}\right] c_2 \longrightarrow c_1 c_2$$

Il s'agit maintenant de préciser les règles de transformation et de décrire les dérivations qu'elles permettent. On l'a dit ci-dessus, leur but est de "diviser" N par D ($2 \leqslant D \leqslant N$). Cette division va consister en une succession de soustractions.

Soit: $N = kD + R \qquad 0 \leqslant R < D$, (N n'étant multiple d'aucun nombre inférieur à D, autre que 1).

la dérivation passera par les mots non terminaux $\left[\begin{array}{c} N - mD \\ \hline D \\ \hline N \end{array}\right]$ $0 \leqslant m \leqslant k$.

- si $R = 0$, la dérivation sera bloquée, pour $m = k$, à $\left[\begin{array}{c} 0 \\ \hline D \\ \hline N \end{array}\right]$;

- si $R \neq 0$, après la $(k + 1)$-ème soustraction de D, la présence de la retenue ($N - (k + 1)D < 0$) nous fera passer au mot non terminal $\left[\begin{array}{c} N \\ \hline D + 1 \\ \hline N \end{array}\right]$

Distinguons donc les différentes étapes et les différents cas suivants:

II. 1. Début de soustraction.

Supposons que nous ayons dérivé de S le mot $\left[\begin{array}{c} N - mD \\ \hline D \\ \hline N' \end{array}\right]$, $0 \leqslant m \leqslant k$, avec:

$$N - mD = r_i \quad \ldots \quad r_1 .$$

A. $r_1 \neq 0$ (donc à fortiori $(N - mD) > 0$)

$$\cdot \left[\begin{array}{c} r_1 \\ \hline d_1 \\ \hline c_1 \end{array}\right] \# \longrightarrow \left[\begin{array}{c} r_1 \\ \hline d'_1 \\ \hline c_1 \end{array}\right] \#$$

Les chiffres r_1, d_1, $_1$, sont quelconques, seule existe la restriction signalée, $r_1 \neq 0$. Cette remarque vaut pour toutes les autres règles.

B. $r_1 = 0$: il s'agit de vérifier si $(N - mD)$ n'est pas nul.

$$(1) \cdot \left[\begin{array}{c} 0 \\ \hline d_1 \\ \hline c_1 \end{array}\right] \# \longrightarrow \left[\begin{array}{c} 0' \\ \hline d_1 \\ \hline c_1 \end{array}\right] \#$$

Nous nous intéressons uniquement aux chiffres de la ligne supérieure, aussi les règles précédentes peuvent, sans aucune confusion, se mettre sous la forme: $[\underline{0}] \# \longrightarrow [\underline{0'}] \#$

étant bien entendu que les deux chiffres inférieurs sont quelconques (à remarquer toutefois qu'ils ne peuvent être indicés) et ne sont pas modifiés par ces règles. Cette remarque nous permettra de simplifier l'écriture de certaines règles

$$\cdot [\underline{0}] [\underline{0'}] \longrightarrow [\underline{0'}] [\underline{0}]$$

. Ainsi dans le cas où N = mD on se trouvera arrêté à:

$$\# \left[\begin{array}{c} 0' \\ \hline 0 \\ \hline c_n \end{array}\right] \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline c_{n-1} \end{array}\right] \ldots \left[\begin{array}{c} 0 \\ \hline d_1 \\ \hline c_1 \end{array}\right] \ldots \left[\begin{array}{c} 0 \\ \hline d_1 \\ \hline c_1 \end{array}\right] \#$$

(Tout au plus pourrait-on ajouter de nouveaux indices “'” à la ligne supérieure).

. Si (N - mD) est positif il existe j, $j \leq i$, tel que r_j est distinct de 0, et il faut alors continuer notre division par D par une nouvelle soustraction:

$$\cdot \quad r \neq 0 \quad \left[\underline{r}\right]\left[\underline{0'}\right] \longrightarrow \left[\underline{r}\right]\left[\underline{0''}\right]$$

$$\cdot \quad \left[\underline{0''}\right]\left[\underline{0}\right] \longrightarrow \left[\underline{0}\right]\left[\underline{0''}\right]$$

$$\left[\begin{array}{c} 0'' \\ \hline d_1 \\ \hline c_1 \end{array}\right] \# \longrightarrow \left[\begin{array}{c} 0 \\ \hline d'_1 \\ \hline c_1 \end{array}\right] \# \quad \text{ou plus simplement} \quad \left[\begin{array}{c} 0'' \\ \hline d_1 \\ \hline \end{array}\right] \# \longrightarrow \left[\begin{array}{c} 0 \\ \hline d'_1 \\ \hline \end{array}\right] \#$$

II. 2. Règles de soustraction.

Seules nous intéressent les deux lignes supérieures; la ligne inférieure, n'étant pas modifiée, conserve l'information du nombre N étudié. Nous avons quatre cas à considérer:

A. Sans retenue (sur le chiffre précédent):

$$\cdot \quad r_2 \geqslant d_2 \quad : \quad \left[\begin{array}{c} r_1 \\ \hline d_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 \\ \hline d'_2 \\ \hline \end{array}\right] \longrightarrow \left[\begin{array}{c} r_1 \\ \hline d'_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 - d_2 \\ \hline d_2 \\ \hline \end{array}\right]$$

$$\cdot \quad r_2 < d_2 \quad : \quad \left[\begin{array}{c} r_1 \\ \hline d_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 \\ \hline d'_2 \\ \hline \end{array}\right] \longrightarrow \left[\begin{array}{c} r_1 \\ \hline d''_1 \\ \hline \end{array}\right]\left[\begin{array}{c} (10 + r_2) - d_2 \\ \hline d_2 \\ \hline \end{array}\right]$$

on introduit la retenue (évidemment égale à 1) en utilisant l'indice« " » .

B. Avec retenue:

$$\cdot \quad r_2 > d_2 \quad : \quad \left[\begin{array}{c} r_1 \\ \hline d_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 \\ \hline d''_2 \\ \hline \end{array}\right] \longrightarrow \left[\begin{array}{c} r_1 \\ \hline d'_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 - d_2 - 1 \\ \hline d_2 \\ \hline \end{array}\right]$$

$$\cdot \quad r_2 \leqslant d_2 \quad : \quad \left[\begin{array}{c} r_1 \\ \hline d_1 \\ \hline \end{array}\right]\left[\begin{array}{c} r_2 \\ \hline d''_2 \\ \hline \end{array}\right] \longrightarrow \left[\begin{array}{c} r_1 \\ \hline d''_1 \\ \hline \end{array}\right]\left[\begin{array}{c} (9 + r_2) - d_2 \\ \hline d_2 \\ \hline \end{array}\right]$$

(introduction de la retenue)

II. 3. Fin de soustraction.

Il y a essentiellemnt deux cas à envisager:

A. $N - (m + 1)D \geqslant 0 \qquad (m < k)$

- sans la retenue:

$$r \geqslant d \ : \ \# \left[\begin{array}{c} r \\ \hline d' \end{array}\right] \longrightarrow \# \left[\begin{array}{c} r - d \\ \hline d \\ \hline \end{array}\right]$$

- avec la retenue:

$$r > d \ : \ \# \left[\begin{array}{c} r \\ \hline d'' \\ \hline \end{array}\right] \longrightarrow \# \left[\begin{array}{c} r - d - 1 \\ \hline d \\ \hline \end{array}\right]$$

B. $N - (m + 1)D < 0 \qquad (m = k)$

- sans la retenue:

$$r < d \ : \ \# \left[\begin{array}{c} r \\ \hline d' \\ \hline c \end{array}\right] \longrightarrow \# \left[\begin{array}{c} c \\ \hline d \\ \hline c' \end{array}\right]$$

- avec la retenue:

$$r \leqslant d \ : \ \# \left[\begin{array}{c} r \\ \hline d'' \\ \hline c \end{array}\right] \longrightarrow \# \left[\begin{array}{c} c \\ \hline d \\ \hline c' \end{array}\right]$$

L'indice « ' » qui apparaît à la ligne inférieure, indique que N n'est pas divisible par D; il va permettre de passer à la division par (D + 1) grâce aux règles suivantes.

II. 4. Passage à la division par (D + 1); c'est-à-dire

passage au mot non terminal: $\left[\begin{array}{c} N \\ \hline D + 1 \\ \hline N \end{array}\right]$

$$\left[\begin{array}{c} c_1 \\ \hline d_1 \\ \hline c'_1 \end{array}\right] \left[\begin{array}{c} r \\ \hline d_2 \\ \hline c_2 \end{array}\right] \longrightarrow \left[\begin{array}{c} c_1 \\ \hline d_1 \\ \hline c_1 \end{array}\right] \left[\begin{array}{c} c_2 \\ \hline d_2 \\ \hline c'_2 \end{array}\right]$$

A. $d_1 \neq 9 \qquad . \left[\begin{array}{c} c_1 \\ \hline d_1 \\ \hline c'_1 \end{array}\right] \# \longrightarrow \left[\begin{array}{c} c_1 \\ \hline d_1 + 1 \\ \hline c_1 \end{array}\right] \#$

B. $d_1 = 9$. $\left[\begin{array}{c} c_1 \\ \hline 9 \\ \hline c'_1 \end{array}\right] \# \longrightarrow \left[\begin{array}{c} c_1 \\ \hline 9 \\ \hline c''_1 \end{array}\right] \#$ la présence de l'indice « " » à la ligne inférieure indique que le passage de D à D + 1 fait intervenir la retenue (D se terminant par un 9).

$$. \left[\begin{array}{c} c_1 \\ \hline d \\ \hline c_1 \end{array}\right] \left[\begin{array}{c} c_2 \\ \hline 9 \\ \hline c''_2 \end{array}\right] \longrightarrow \left[\begin{array}{c} c_1 \\ \hline d \\ \hline c''_1 \end{array}\right] \left[\begin{array}{c} c_2 \\ \hline 0 \\ \hline c_2 \end{array}\right]$$

$$. \ d \neq 9 \quad \left[\begin{array}{c} c \\ \hline d \\ \hline c'' \end{array}\right] \longrightarrow \left[\begin{array}{c} c \\ \hline d+1 \\ \hline c \end{array}\right]$$

Il résulte de la description faite que tout nombre premier N dérive de S, selon la grammaire $\mathcal{G}$. Il resterait à voir que la grammaire $\mathcal{G}$ n'engendre pas, ou plutôt "n'accepte pas" d'autres nombres. On remarquera que les règles doivent être appliquées dans l'ordre de leur présentation si l'on veut que la dérivation aboutisse à une phrase. Pour cela on considérera l'apparition et la disparition des indices qui commandent, en quelque sorte, l'enchaînement des étapes.

Ainsi s'achève la démonstration du résultat ennoncé:

$$\mathcal{L}(\mathcal{G}) = \{ N \mid N \text{ nombre premier décimal} \} .$$

REMARQUES

1°)- A ce résultat nous pouvons ajouter le suivant:

Le complémentaire du langage $\mathcal{L}(\mathcal{G})$, c'est-à-dire le langage: $\mathcal{L}' = \{ N' \mid N' \text{ entier naturel décimal non premier} \}$ est un langage CS.

Il suffit de considérer la grammaire $\mathcal{G}$ et d'en remplacer les règles terminales III par les suivantes:

$$. \ d < c \quad \# \left[\begin{array}{c} 0 \\ \hline d \\ \hline c \end{array}\right] \longrightarrow \# \ c$$

$$\cdot\ c_1 \left[\begin{array}{c} 0 \\ \hline d_2 \\ \hline c_2 \end{array}\right] \longrightarrow c_1 c_2$$

en ajoutant $\# S \# \longrightarrow \# 0 \#$ et $\# S \# \longrightarrow \# 1 \#$.

Le lecteur vérifiera ce résultat facilement en remarquant que N n'est pas premier si et seulement s'il existe D, $D < N$ (on peut préciser $D \leq \frac{N}{2}$) tel que $N = kD$. Dans une dérivation selon la grammaire $\mathcal{G}$, essayant d'engendrer N, on aboutit au mot $\left[\begin{array}{c} 0 \\ \hline D \\ \hline N \end{array}\right]$. En posant:

$N = c_n \dots c_1$ et $D = d_1 \dots d_1$

on est sûr que l'on a $d_n < c_n$ (car $N \geqslant 2D$) d'où la restriction imposée ci-dessus à la première règle terminale applicable. Elle était nécessaire, car si N est premier, la grammaire $\mathcal{G}$ nous permet d'aboutir au mot $\left[\begin{array}{c} N \\ \hline N \\ \hline N \end{array}\right]$ et en continuant d'appliquer les règles transformation, au mot $\left[\begin{array}{c} 0 \\ \hline N \\ \hline N \end{array}\right]$, mais alors $d_n = c_n$ et notre nouvelle grammaire "n'accepte pas" N.

2°) Nous allons donner une nouvelle grammaire CS engendrant le langage:

$$\mathcal{L}(p) = \left\{ |^p \mid p \in \overset{=}{\mathbb{N}},\ p \text{ premier} \right\} ,$$

Auparavant, donnons un ensemble $\mathcal{R}_1$ de règles CS permettant de passer de l'écriture décimale d'un entier naturel p à son écriture "en bâtons".

. Règles CS de passage de $\# p \#$ à $\# |^p \#$.

Soit $p = c_n c_{n-1} \dots c_2 c_1$.

Nous allons distinguer 3 cas suivant la valeur du chiffre des unités.

- 1er cas: $c > 1$. $c \# \longrightarrow (c - 1) | \#$

. $c| \longrightarrow (c - 1) ||$

- $2^{\text{ème}}$ cas $C = 1$ $\left.\begin{array}{l} .\ d\,1\,\# \longrightarrow d\,0\,|\,\# \\ .\ d\,1\,| \longrightarrow d\,0\,|\,| \end{array}\right\}$ d est un chiffre quelconque.

règle de fin de transformation $.\ \#\,1\,| \longrightarrow \#\,|\,|$

cas particulier $.\ \#\,1\,\# \longrightarrow \#\,|\,\#$

- 3ème cas: $C = 0$ $.\ 0\,\# \longrightarrow 0'\,\#$

$.\ 0\,| \longrightarrow 0'\,|$

$.\ 0\,0' \longrightarrow 0'\,0$

$d > 1$ $.\ d\,0' \longrightarrow (d - 1)9\,|'$

$d = 1 \begin{cases} .\ u\,1\,0' \longrightarrow u\,0\,9\,|' \\ .\ \#\,1\,0' \longrightarrow \#\,9\,|' \end{cases}$ u est un chiffre quelconque

$.\ |'\,0 \longrightarrow 9\,|'$

$.\ |'\,\# \longrightarrow |\,\#$

$.\ |'\,| \longrightarrow |\,|$

On remarquera que la dérivation

$$\#\,p\,\# \Longrightarrow \#\,|^{p}\,\#$$

est définie d'une façon unique; on passera par les étapes:

$\#\,p\,\# \Longrightarrow \#\,(p - 1)\,|\,\# \Longrightarrow \dots\,\#\,q\,|^{p-q}\,\# \longrightarrow \dots \longrightarrow \#\,|^{p}\,\#$,

le passage de q à (q - 1) ($1 < q \lesssim p$) exige, dans le cas où le chiffre des unités de q est nul, l'intervention d'une retenue.

Nous sommes maintenant en mesure de définir une grammaire engendrant $\mathcal{L}(p)$:

$$\mathcal{G}' = (\mathcal{V}', \{\,|\,\}, S, \#, \mathcal{R}')$$

- $\mathcal{V}' = \mathcal{V} \cup \{0', |'\}$, $\mathcal{V}$ étant le vocabulaire terminal et non terminal de la grammaire $\mathcal{G}$ étudiée ci-dessus.

- $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_1$, $\mathcal{R}$ étant l'ensemble des règles de la grammaire $\mathcal{G}$ et $\mathcal{R}_1$ l'ensemble des règles de passage décrites ci-dessus.

On a bien: $\mathcal{L}(\mathcal{G}') = \mathcal{L}(p)$

En conclusion nous allons énoncer des résultats permettant de comparer la classe des langages CS à deux autres classes qui généralisent celle des langages CF.

- Langages CF matricielles.

Une grammaire matricielle est un couple $[\mathcal{G},\mathcal{M}]$ où $\mathcal{G}$ est une grammaire de constituants.

$$\mathcal{G} = (\mathcal{V}, \mathcal{V}_T, S, \mathcal{R})$$

et $\mathcal{M}$ une partie finie du monoïde libre $\mathcal{R}^*$. $[\mathcal{G},\mathcal{M}]$ sera dite grammaire matricielle CF, CS, etc., si $\mathcal{G}$ est respectivement une grammaire CF, CS, etc... Le langage engendré par la grammaire matricielle $[\mathcal{G},\mathcal{M}]$ est l'ensemble:

$$\mathcal{L}[\mathcal{G},\mathcal{M}] = \{\hat{x} \mid \hat{x} \in \mathcal{V}_T^* ; \exists \mathfrak{p} \in \mathcal{M}^* : S \xRightarrow[\mathcal{G}]{\mathfrak{p}} \hat{x}\}$$

des mots sur le vocabulaire $\mathcal{V}_T$ qui dérivent de S par une suite de règles de $\mathcal{R}$ qui est un produit de juxtaposition de mots de $\mathcal{M}$. (Autrement dit on applique d'abord dans leur ordre les règles d'un premier mot $\mathfrak{p}_1 \in \mathcal{M}$; puis on passe à un second mot $\mathfrak{p}_2$, dont on doit pouvoir appliquer la première "lettre", qui est une règle de $\mathcal{R}$, puis la seconde etc...).

J. P. Benzécri a prouvé [2] que :

. tout langage engendré par une grammaire CF matricielle est CS...

On a plus: tout langage CS matriciel est CS.

- Langages CFP (cf. § 2 au début)

Nous avons prouvé (travail à paraître) que:

. tout langage CFP est CS.

BIBLIOGRAPHIE

-o-O-o-

[1] J. P. BENZECRI (Rennes, février 1965) Structures algèbriques et constituants non connexes dans les grammaires.

[2] J. P. BENZECRI (Paris, 1966) Grammaires matricielles

[3] N. CHOMSKY, Formal Properties of Grammars, dans Handbook of Mathematical Psychology, vol. II (Ed. by D. Luce, E. Busch, E. Galanter) 1963, pp. 323-418.

[4] J. FRIANT (Thèse de 3° cycle, Paris, 1966) Les langages CS.

[5] J. FRIANT (Paris 1966) Nombres premiers et langages CS.

[6] R. GUEDJ (Thèse de 3° cycle, Paris, 1966) Grammaires de Constituants Généraux.

[7] S. Y. KURODA (Octobre 1963), Classes of Languages and Linear bounded autamata.

[8] P. S. LANDWEBER, Three theorems on phrase structure grammars of type 1, Information and Control, t. 6, 1963, p. 137-146.

LES GRAMMAIRES DE CONSTITUANTS GENERAUX.

par R. D. Guedj.

Il est deux faits linguistiques d'une grande importance qui nous ont conduits à définir un nouveau type de grammaire formelle, les grammaires de constituants généraux (GCG):

- la présence dans les langues naturelles de mots non connexes
- la distinction entre le niveau de la composition et le niveau de l'expression dans la génération ou la reconnaissance d'une phrase.

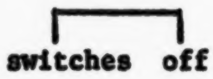
0.1 . Mots non connexes.

Si nous voulons analyser une phrase aussi simple que

" the boy switches the light off "

En tout premier lieu nous pouvons dire que cette phrase est formée, dans l'ordre, d'un article, d'un substantif, d'un verbe, d'un article, d'un substantif, d'une postposition; Cependant il est bien clair que le verbe switches et la postposition off sont intimement liés pour former le verbe switches off, un tel mot sera appelé mot non connexe à deux insertions.

Les GCG vont nous permettre d'accepter des mots non connexes à plusieurs insertions comme des entités; c'est ainsi que l'on représente un mot à 2 insertions par le lien qui unit les deux parties qui le constituent:



(que l'on appellera aussi peigne à deux dents).

0.2. Expression et Composition.

Il est un fait essentiel sur lequel nous nous permettrons d'insister dès à présent. Dans la génération ou la reconnaissance d'une phrase nous distinguons deux niveaux: le niveau de l'expression et le niveau de la composition. Tout d'abord un exemple choisi hors du domaine linguistique nous permettra de rendre plus clair cette distinction. Lorsqu'une entreprise désire construire un immeuble de n appartements elle a, au préalable, constitué un stock de briques, de sable, de tuiles, de verre, de bois,... c'est ce stock de matériaux que nous appelons la composition de l'immeuble. Puis l'architecte utilise les éléments qui constituent ce stock pour bâtir un immeuble. Cependant il pourra disposer les appartements de manières différentes, et c'est la disposition des appartements que nous appelons l'expression de l'immeuble. Ainsi deux immeubles peuvent différer au niveau de la composition (par la nature ou l'importance du stock) et de plus ayant même compostions il peuvent différer par l'expression (par la disposition des appartements). Revenons à notre objet, en linguistique ces deux niveaux apparaissent clairement. Au niveau de la compostion nous nous bornons à énoncer les types syntaxiques nécessaires à la formation d'une phrase (remarquons que l'ordre dans lequel nous les disposons a peu d'importance ici).
Au niveau de l'expression, nous commençons par donner des valeurs aux différents types syntaxiques de la liste fournie par la composition (ces valeurs pouvant être des mots à plusieurs insertions), ensuite nous indiquons la disposition relative de ces mots. Dans le cas de mots non connexes il s'agit d'imbriquer les peignes qui représentent ces mots, avec l'aide des "morphismes de peignes".

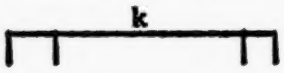
Dans cet article nous donnons la définition de la catégorie SEG, des arbres biordonnés et des GCG, cette dernière définition suppose connues les notions fondamentales de la théorie des catégories. Ensuite nous donnons une propriété algébrique des langages engendrés par les GCG, qui nous permettra de les comparer aux types de grammaires déjà existants.

§ 1. CATEGORIE SEG.

1.1. Simplexe. Soit $k \in \mathbb{N}$. On appellera simplexe k (noté $]k]$):

$$]k] = \{1, 2, \dots, k\},$$

on le représente aussi par le peigne à k dents

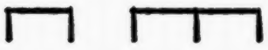


1.2. Simplexe segmenté. On appelle simplexe segmenté une suite de simplexes.

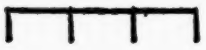
On appelle produit des simplexes segmentés $S_1, \dots, S_n$ le simplexe segmenté représenté par le peigne obtenu en juxtaposant de gauche à droite les peignes qui représentent successivement $S_1 \dots, S_n$ (que l'on note $S = \overset{*}{i = 1 \dots n} S_i$)

Ex:

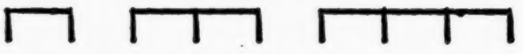
S_1 :



S_2 :



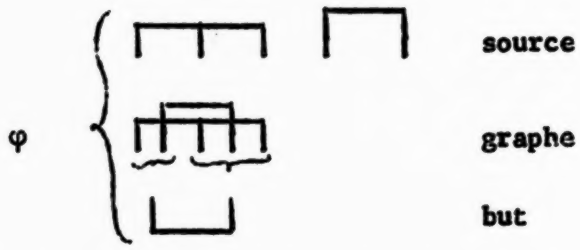
$S_1 * S_2$:



1.3. Morphismes de peignes.

Soit un simplexe segmenté (source) un morphisme de peignes assemble les peignes qui constituent ce simplexe segmenté (graphe pour construire un nouveau simplexe segmenté (but). Il nous faut donc préciser comment les peignes de la source sont assemblés pour former les peignes du but.

Ex: le morphisme φ :



la source . φ est le simplexe segmenté composé dans l'ordre d'un peigne n_1 à trois dents et d'un peigne n_2 à deux dents.

le graphe de φ indique que l'on insère la première dent du peigne n_2 entre la $1^{ère}$ et la $2^{ème}$ dent du peigne n_1 et que l'on insère la deuxième dent du peigne n_2 entre la $2^{ème}$ et la $3^{ème}$ dent du peigne n_1 .

le but de φ est un peigne à 2 dents, la première est formée, dans l'ordre, par la $1^{ère}$ de n_1 et la $1^{ère}$ de n_2 , la deuxième par la $2^{ème}$ de n_1 , la $2^{ème}$ de n_2 et la $3^{ème}$ de n_1 (on retourne le peigne du but pour marquer plus clairement la manière de l'obtenir).

1.4. Composition de morphismes.

Soit deux morphismes φ et π tels que source [illegible] φ.

on définit le morphisme (π ∘ φ) de la manière suivante:

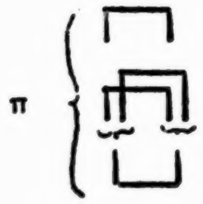
source (π ∘ φ) = source φ

but (π ∘ φ) = but φ

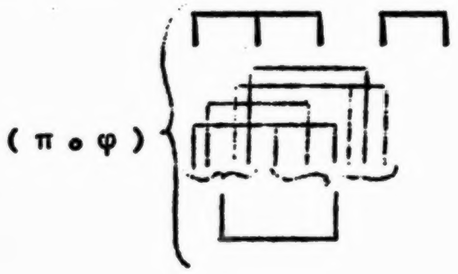
et graphe de (π ∘ φ) est déduit de graphe φ et graphe π de la manière que rendra clair l'exemple suivant:

φ est le morphisme défini plus haut

π est le morphisme



alors ($\pi \circ \varphi$) est le morphisme:



1.5. Catégorie SEG.

La catégorie SEG est une catégorie à produit direct dont les objets sont les simplexes segmentés, les flèches sont les morphismes de peignes et la composition des flèches est la composition des morphismes de peignes.

§ 2. ARBRES BIORDONNÉS.

2.1. Définition.

Un arbre biordonné est un ensemble $\mathcal{A}$ fini muni de deux relations d'ordre: $\mathcal{H}$ (ordre hiérarchique) et $\mathcal{S}$ (ordre séquentiel) assujetties aux conditions suivantes:

A_1 : le prédécesseur immédiat pour $\mathcal{H}$ de tout élément de $\mathcal{A}$ lorsqu'il existe, est unique.

A_2 : $\mathcal{S}$ est une relation d'ordre total.

A_3 : $\forall$ A, B $\in \mathcal{A}$; (A,B) $\Longrightarrow$ $\mathcal{S}$(A,B)

A_4 : $\forall$ A, A', B, B' $\in \mathcal{A}$,

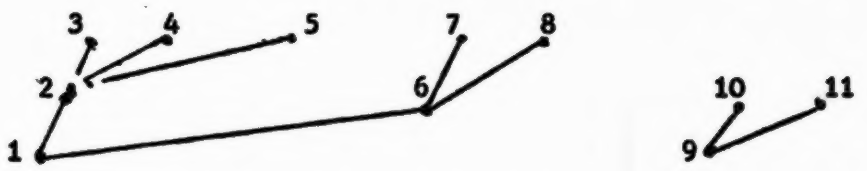
$$\left(\mathcal{S}(A,B),\ \mathcal{H}(A,A'),\ \mathcal{H}(B,B')\right) \Longrightarrow \mathcal{S}(A',B')$$

Un élément sans prédécesseur pour $\mathcal{H}$ est appelé sommet de $\mathcal{A}$, un arbre biordonné qui admet plusieurs sommets est dit non connexe.

2.2. Listes.

Etant donné un ensemble fini (l'alphabet), on appelle liste sur Λ le couple ($\mathcal{A}$, $\mathcal{E}$) d'un arbre biordonné et d'une application étiquetage $\mathcal{E}$ de $\mathcal{A}$ dans le monoïde libre construit sur Λ.

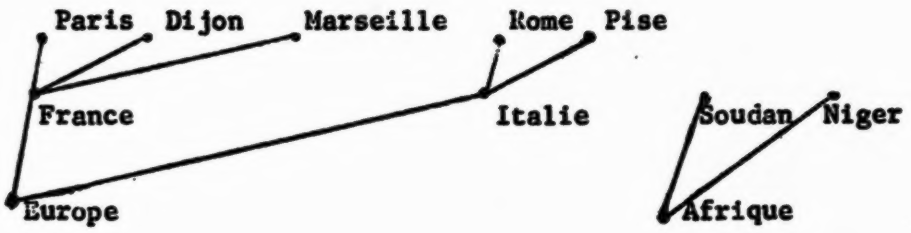
Exemple: Soit l'arbre biordonné suivant, les nombres attachés à chaque noeud sont leurs numéros dans l'ordre $\mathcal{S}$:



et soit l'alphabet Λ = $\{$Europe, France, Italie, Afrique, Soudan, Niger, Rome, Pise, Dijon, Paris, Marseille$\}$

et $\mathcal{E}(1)$ = Europe, $\mathcal{E}(2)$ = France, $\mathcal{E}(3)$ = Paris, $\mathcal{E}(4)$ = Dijon, $\mathcal{E}(5)$ = Marseille, $\mathcal{E}(6)$ = Italie, $\mathcal{E}(7)$ = Rome, $\mathcal{E}(8)$ = Pise, $\mathcal{E}(9)$ = Afrique, $\mathcal{E}(10)$ = Soudan, $\mathcal{E}(11)$ = Niger

On forme la liste ($\mathcal{A}$, $\mathcal{E}$) :



§ 3. GRAMMAIRES DE CONSTITUANTS GENERAUX.

Une GCG est la donnée du 5-upple suivant:

$$\mathcal{G} = \{ \Lambda_N , \mathcal{R}, \mathcal{T} ; [\Lambda_T] , \xi , \alpha' \}$$

les 3 premières composantes constituent la composition et les trois dernières contituent l'expression.

3.1. Composition: $C = (\Lambda_N , \mathcal{R}, \mathcal{T})$.

Λ_N : ensemble fini appelé alphabet nonterminal ou alphabet des catégories syntaxiques (dont les éléments sont notés par une capitale latine), Λ_N comprend un élément distingué, l'élément initial S.

$\mathcal{R}$: ensemble fini de règles syntaxiques r, muni de deux applications α source et β but :

$$\mathcal{R} \xrightarrow{\alpha} \Lambda_N^* \quad \text{et} \quad \mathcal{R} \xrightarrow{\beta} \Lambda_N$$

une règle r sera notée aussi:

$$r : \beta(r) \longrightarrow \alpha(r) \text{ ou } r_{\beta(r)}^{\alpha(r)}$$

$\mathcal{T}$: ensemble fini de règles lexicales t, muni d'une application β' but,

$$\beta' : \mathcal{T} \xrightarrow{\beta'} \Lambda_N .$$

$\forall A \in \Lambda_N$ on définit l'ensemble $\mathcal{T}_A$:

$$\mathcal{T}_A = \{ t \in \mathcal{T} \mid \beta'(t) = A \}$$

éventuellement un $\mathcal{T}_A$ peut être vide.

Soit Λ_N' l'image par β' de $\mathcal{T}$, c'est l'ensemble des catégories syntaxiques auxquelles on fera correspondre dans l'expression, à l'aide de l'application source α' de $\mathcal{T}$, une expression terminale.

Disons maintenant quelles sont les constructions élaborées au niveau de la composition.

On définit une catégorie avec produit tensoriel SYN qui a pour objet les mots du monoïde Λ_N^* , le produit des objets n'étant autre que le produit de juxtaposition du monoïde. Les morphismes de SYN sont construite à partir du système $\mathcal{R}$ de morphismes générateurs.

Pour $B \in \Lambda_N$ et $\hat{A} \in \Lambda_N^*$, Soit $Hom_{SYN}(\hat{A},B)$ l'ensemble des morphismes de SYN ayant $\hat{A}$ pour source et B pour but, cet ensemble n'est autre que l'ensemble des arbres biordonnés de sommet B et de base $\hat{A}$, construits en prenant pour liens les règles de $\mathcal{R}$ (à chaque noeud de l'arbre correspond une lettre de Λ_N , à chaque noeud non situé à la base correspond une règle de $\mathcal{R}$ dont le but est la lettre du noeud et dont la source est le mot de Λ_N^* formé, dans l'ordre séquentiel, par la suite des lettres des noeuds immédiatement inférieurs).

Un élément $s_B^{\hat{A}} \in Hom_{SYN}(\hat{A},B)$ est appelé un schéma de syntagme de type B (construit à partir de $\hat{A}$). Dans la terminologie classique $s_B^{\hat{A}}$ serait la donnée d'une certaine B-dérivation de $\hat{A}$.

On définit ensuite un syntagme composé de type B, $(s_B^{\hat{A}} , \hat{t})$ comme un couple d'un schéma de syntagme de type B, et d'un mot $\hat{t} \in \mathcal{T}^*$ tel que $\beta'(\hat{t}) = \hat{A}$, c'est-à-dire qu'un schéma de syntagme de type B ne peut permettre la construction d'un syntagme composé de type B que si $\hat{A} \in (\Lambda'_N)^*$.

Un syntagme composé de type B pourra aussi être notée $s_B^{\hat{A}}(\hat{t})$.

3.2. Expression. $E = \left(\left[\Lambda_T \right] , \xi, \alpha' \right)$.

Nous supposerons ici que la catégorie d'expression est toujours SEG.

$\left[\Lambda_T \right] = \left[\Lambda_T^1 , \dots , \Lambda_T^n \right]$ alphabet terminal gradué,

Λ_T^i : ensemble des mots terminaux à i-insertions. On considère le SEG-oïde (ou échelle libre) construit sur $\left[\Lambda_T \right]$.

ξ : foncteur de SYN dans SEG commutant ave le produit tensoriel. On désigne par $\xi(A)$ l'entier strictement positif, nombre d'insertions de l'expression des syntagmes de catégorie syntaxique A. On désigne par $\xi(r)$ le morphisme de SEG correspondant à la règle r : $\xi(r)$ aura pour but le peigne à $\xi(\psi(r))$ dents et pour source la liste de peignes à $\xi(A_i)$ dents (si $\alpha(r) = A_1 \ldots A_n$).

α' : est une famille d'applications $\left\{ \alpha'_A \right\}_{A \in \Lambda_N}$ qui envoient $\mathcal{C}_A$ dans $(\Lambda^*_T)^{\xi(A)}$, c'est-à-dire que à chaque terme on fait correspondre un mot ayant le nombre d'insertions convenable...

On définit maintenant un syntagme exprimé. Soit un syntagme composé $s_B^{\hat{A}}(\hat{t})$, α' associe à $\hat{t}$, considéré comme liste de termes t_i, une liste mots à plusieurs insertions $\alpha'(t_i)$

$$\hat{t} \rightsquigarrow\xrightarrow{\alpha'} \left\{ \alpha'(t_i) \right\} = \alpha'(\hat{t})$$

avec $\forall\ i \quad \alpha'(t_i) \in (\Lambda^*_T)^{\xi(\beta'(t_i))}$

sur la liste $\left\{ \alpha'(t_i) \right\}$ opère le morphisme de SEG, $\xi(s_B^{\hat{A}})$ le résultat de cette opération sera un syntagme exprimé de type B, expression du syntagme composé $s_B^{\hat{A}}(\hat{t})$. On pourra écrire ce syntagme ~~exprimé~~ $\xi(s_B^{\hat{A}})\left(\alpha'(\hat{t})\right)$. On note ξ_B l'ensemble des syntagmes exprimés de type B, en particulier ξ_S sera le langage engendré par la grammaire de constituants généraux

$L(\mathcal{G}) = \xi_S$

et si $\xi(S) = 1$ on a $t(\mathcal{G}) \subset \Lambda^*_T$

On remarque qu'un même syntagme exprimé peut être l'expression de plus syntagmes composés, c'est pourquoi on définit l'ambiguité de structure d'un syntagme exprimé $\xi(s_B^{\hat{A}})\left(\alpha'(t)\right)$ comme le cardinal de l'ensemble:

$$\left\{ \left(s_B^{\hat{A}^i}, \hat{t}^i\right) \mid \xi\left(s_B^{\hat{A}}\right)' \alpha'(\hat{t}^i) = \xi\left(s_B^{\hat{A}}\right)\left(\alpha'(\hat{t})\right) \right\}$$

si l'on caractérise les règles de types CF comme ayant un membre de gauche appartenant à Λ_N et non Λ^*_N, on peut considérer les GCG comme des grammaires de type CF plus générales puisque $\forall\, r \in \mathcal{R}$ et $t \in \mathcal{T}$, $\beta(r) \in \Lambda_N$ et $\beta'(t) \in \Lambda_N$.

- Une GCG soumise aux restrictions:

α - seul $\Lambda_T^1 \neq \emptyset$

β - $\xi(S) = 1$

est appelée Grammaire Context Free à Peignes (GCFP)

- Une GCG (resp. GCFP) dans laquelle $\forall\, r \in \mathcal{R}$, $\xi(r)$ est un morphisme de SEG tel que tout peigne de la source $\xi(\ \alpha(r)\ )$ est utilisé une fois et une seule dans le graphe, est appelée GCG linéaire (GCGL) (resp. GCFPL).

Nous allons montrer que les GCFP sont une généralisation des grammaires CF. Soit une GCF $\mathcal{G}' = \{ V', V_T, \mathcal{R}, S \}$. On sait qu'à tout GCF on peut faire correspondre une GCF syntaxique équivalente. Une grammaire syntaxique est une grammaire dans laquelle l'ensemble $\mathcal{R}$ des règles est la réunion disjointes des ensembles $\mathcal{R}_T$ et $\mathcal{R}_N$, définis ainsi:

$\mathcal{R}_T = \{ r\colon A \longrightarrow \hat{B} \mid \hat{B} \in (V_T)^* \}$ et $\mathcal{R}_N = \{ r\colon A \longrightarrow \hat{B} \mid \hat{B} \in (V - \{V_T\})^* \}$

Ainsi soit une GCGL $\mathcal{G}$ dans laquelle nous faisons, au niveau de l'expression, la restriction γ

γ - $\forall\, A \in \Lambda_N \quad \xi(A) = 1$

(On remarque que γ est plus puissante que β)

la restriction γ et le fait que la grammaire $\mathcal{G}$ soit linéaire impliquent que $\forall\, r \in \mathcal{R}$, $\xi(r)$ est un produit de juxtaposition (au sens monoïde).

Et d'après α, puisque l'alphabet $[\ \Lambda_T\ ]$ n'est plus un alphabet gradué, l'échelle libre sur $[\Lambda_T\ ]$ se réduit au monoïde libre sur Λ_T^1.

Et aini à tout règle $r_N \in \mathcal{R}_N$ (de $\mathcal{G}'$) $r_N\colon B \longrightarrow \hat{A}$ on fait correspondre d'une manière unique une règle $r \in \mathcal{R}$ (de $\mathcal{G}$)

$r\ :\ B \longrightarrow \hat{A}$ (et réciproquement).

A toute règle $r_T \in \mathcal{R}_T$ (de $\mathcal{G}'$) $\cdot\, r_T : B \longrightarrow \varphi \in V_T^*$ on fait correspondre d'une manière unique une règle $t \in \mathcal{T}$ (de $\mathcal{G}$) $t : B \longrightarrow$ telle que au niveau de l'expression $\alpha'(t) = \hat{\varphi}$ (et réciproquement).

Lemme

Ainsi à toute GCF $\mathcal{G}'$ on peut faire correspondre une GCFPL $\mathcal{G}$, soumises aux deux restrictions α et γ, telle que $\mathcal{G}'$ et $\mathcal{G}$ soient fortement équivalentes.

Nous allons maintenant donner l'exemple d'une GCFP (de peu d'intérêt linguistique puisqu'elle n'engendre qu'une phrase).

Exemple:

Soit la GCFP $\mathcal{G}_A = \left\{ \Lambda_N , \mathcal{R} , \mathcal{T} ; \Lambda_{T_A} , \xi_A , \alpha'_A \right\}$.

Données de Composition.

- $\Lambda_N = \left\{ S , Art , N , V , S_N \right\}$
- $\mathcal{R} = \left\{ r_o , r_1 \right\}$

$r_o : S \longrightarrow S_N . V . S_N$, $S_N \longrightarrow Art . N$

- $\mathcal{T} = \left\{ t_o , t_1 , t_2 , t_3 \right\}$

$t_o : V \longrightarrow$, $t_1 : N \longrightarrow$, $t_2 : N \longrightarrow$, $t_3 : Art \longrightarrow$

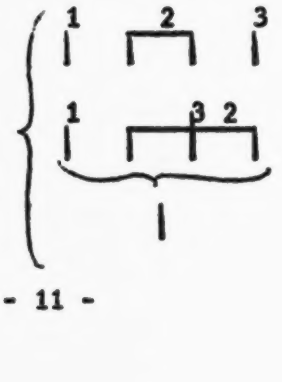
Données d'expression.

- $\Lambda_{T_A} = \left\{ \text{the, off, switches, boy, light} \right\}$
- ξ_A est défini par :

$\xi_A(S) = 1$, $\xi_A(V) = 2$, $\xi_A(N) = 1$, $\xi_A(Art) = 1$.

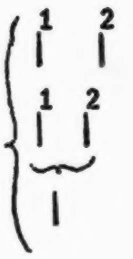
$\xi_A(r_o) = \psi$

$\psi =$



$\xi_A(r_1) = \varphi$

φ



- α'_A est défini par:

  $\alpha'_A(t_o)$ = switches-off

  $\alpha'_A(t_1)$ = boy, $\alpha'_A(t_2)$ = lights, $\alpha'_A(t_3)$ = the

- le seul syntagme composé construit avec $\mathcal{G}_A$ est:

  $(s = s_S^{\text{Art.N.V.Art.N}} \; ; \; \hat{t} = t_3 t_1 t_o t_3 t_2)$.

et le seul syngagme exprimé est: $\xi_A(s)\left\{ \alpha'_A(\hat{t}) \right\}$

c'est-à-dire la phrase: the boy switches the light off.

§ 4. UNE PROPRIÉTÉ ALGEBRIQUE DES GCF.

Soit une GCG $\mathcal{G}$ telle que l'alphabet terminal est réduit à un seul terme x qui a une insertion: $\Lambda_T = \{ x \}$

On démontre le théorème suivant:

Théorème. Tout langage L sur un alphabet $\Lambda_T = \{ x \}$ contient soit une progression arithmétique sur x, soit une progression géométrique de la forme:

$$L = \left\{ x^{W + uV^k} \mid u, V, W, \text{ fixés} \in \mathbb{N}, \text{ avec } u \text{ et } V \neq 0 \text{ et } k \in \mathbb{N} \right\}$$

c'est ainsi que l'on démontre que le langage L(!),

$$L(!) = \left\{ x^{n!} \mid n \in \mathbb{N} \right\}$$

n'est pas du type GCG, alors que Friant montre que L(!) est engendré par une grammaire CS.

C'est pourquoi on peut donner la classification suivante:

CF $\subset$ GFP $\subset$ CS

les inclusions sont toutes strictes.

BIBLIOGRAPHIE

- o - 0 - o -

J. P. Benzécri, Modo Imago Nati (Inst. Stat. Univ. Paris, 1966)

J. P. Benzécri, Linguistique Mathématique, Algèbre des constituants non connexes (Inst. Stat. Univ. Paris, 1966).

J. Friant, Thèse de 3° cycle: Les langages CS (Inst. Stat. Univ. Paris, 1966).

R. D. Guedj, Thèse de 3° cycle: Les grammaires de constituants généraux (Inst. Stat. Univ. Paris, 1966)

38

Research Group for
Quantitative Linguistics
Fack
Stockholm 40
SWEDEN

KVAL PM 339
June 19, 1967

The Entropy of Recursive Markov Processes

By

BENNY BRODDA

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

THE ENTROPY OF RECURSIVE MARKOV PROCESSES

By

BENNY BRODDA

KVAL, Fack, Stockholm 40, Sweden

Summary

The aim of this communication is to obtain an explicit formula for calculating the entropy of a source which behaves in accordance with the rules of an arbitrary Phrase Structure Grammar, in which relative probabilities are attached to the rules in the grammar. With this aim in mind we introduce an alternative definition of the concept of a PSG as a set of self-embedded (recursive) Finite State Grammars; when the probabilities are taken into account in such a grammar we call it a Recursive Markov Process.

1. In the first section we give a more detailed definition of what kind of Markov Processes we are going to generalize later on (in sec. 3), and we also outline the concept of entropy in an ordinary Markov source. More details of information may be found, e.g., in Khinchins "Mathematical Foundations of Information Theory", N.Y., 1957, or "Information Theory" by R. Ash, N.Y., 1965.

A Markov Grammar is defined as a Markov source with the following properties:

Assume that there are n + 1 states, say S_0, S_1, ..., S_n, in the source. S_0 is defined as the initial state and S_n is defined as the final state and the other states are called intermediate states. We shall, of course, also have a transition matrix, $M = (p_{ij})$, containing the transition probabilities of the source.

a) A transition from state S_i to state S_k is always accompanied by a production of a (non-zero) letter a_{ik} from a given finite alphabet. Transition to different states from one given state always produce different letters.

b) From the initial state, S_0, direct or indirect transitions should be possible to any other state in the source. From no state is a transition to S_0 allowed.

c) From any state, direct or indirect transitions to the final state S_n should be possible. From S_n no transition is allowed to any other state (S_n is an "absorbing state").

The work reported in this paper has been sponsored by Humanistiska forskningsrådet, Tekniska forskningsrådet and Riksbankens Jubileumsfond, Stockholm, Sweden.

A (grammatical) *sentence* should now be defined as the (left-to-right) concatenation of the letters produced by the source, when passing from the initial state to the final state.

The *length* of a sentence is defined as the number of letters in the sentence. To simplify matters without dropping much of generality we also require that

d) The greatest common divisor for all the possible lengths of sentences is = 1 (i.e., the source becomes an aperiodic source, if it is short-circuited by identifying the final and initial states).

With the properties a - d above, the source obtained by identifying the final and initial states is an indecomposable, ergodic Markov process (cf. Feller, "Probability Theory and Its Applications", ch. 15, N.Y., 1950).

In the transition matrix M for a Markov grammar of our type all elements in the first column are zero, and in the last row all elements are zero except the last one which is = 1. For a given Markov grammar we define the *uncertainty* or *entropy*, H_i, for each state S_i, $i = 0, 1, \ldots, n$, as:

$$H_i = -\sum_{j=0}^{n} p_{ij} \log p_{ij}; \; i = 1, 2, \ldots, n.$$

We also define the entropy, H or H(M), for the grammar as

$$(1). \quad H = \sum_{i=0}^{n=1} x_i H_i$$

where $x = (x_0, x_2, \ldots, x_{n-1})$ is defined as the stationary distribution for the source obtained when S_0 and S_n are identified; thus x is defined as the (unique) solution to the set of simultaneous equations

$$(2) \quad \begin{aligned} xM_1 &= x \\ x_0 + x_1 + \ldots + x_{n-1} &= 1 \end{aligned}$$

where M_1 is formed by shifting the last and first columns and then omitting the last row and column. The *mean sentence length*, μ, of the set of grammatical sentences can now be easily calculated as

$$(3) \qquad \mu = 1/x_0$$

(cf. Feller, op. cit.)

2. Embedded Grammars

We now assume that we have two Markov grammars, M and M_1, with states $S_0, S_1, \ldots, S_n$, and $T_0, T_1, \ldots, T_m$, respectively, where S_0 and S_n, T_0 and T_m are the corresponding initial and final states. Now consider two states S_i and S_k in the grammar M; assume that the corresponding transition probability is = p_{ik}. We now transform the grammar, M_1, into a new one, M_1', by embedding the grammar M_2 in M_1 between the states S_i and S_k, an operation which is performed by identifying the states T_0 and T_m with the states S_i and S_k respectively. Or, to be more precise, assume that in the grammar M_1 the transitions to the states T_j, $j \geq 1$, has the probabilities q_{0j}. Then, in the grammar M', transitions to a state T_j from the state S_i will take place with the probability = $p_{ik}q_{0j}$. A return to the state S_k in the "main" grammar from an intermediate state T_j in M_1 takes place with the probability q_{jm}.

With the conditions above fulfilled, we propose that the entropy for the composed grammar be calculated according to the formula:

$$(4) \qquad H(M') = \frac{H(M) + x_i p_{ik} \cdot \mu_1 \cdot H(M_1)}{1 + x_i p_{ik} (\mu_1 - 1)}$$

where H(M) is the entropy of the grammar M when there is an ordinary connection (with probability p_{ik}) between the states S_i and S_k, and where x_i is the inherent probability of being in the state S_i under the same conditions. μ_1 is the mean sentence length of the sentences produced by the grammar M_1 alone. (It is quite natural that this number appears as a weight in the formula, since if one is producing a sentence according to the grammar M and arrives at the state S_i and from there "dives" into the grammar M_1, then μ_1 is the expected waiting time for emerging again in the main grammar M.) The factor $x_i p_{ik}$ may be interpreted as the combined probability of ever arriving at S_i and there choosing the path over to M_1 (you may, of course, choose quite another path from S_i).

The proof of formula (4) is very straightforward, once the premises according to the above have been given, and we omit it here, as it does not give much extra insight to the theory. The formula may be extended to the case when there are more than one sub-grammar embedded in the grammar M', by adding similar terms as the one standing to the right in the numerator and the denominator. The important thing here is that the factors of the type $x_i p_{ik}$ depend only on the probability matrix for the grammar M and are dependent of the sub-grammars involved.

3. Recursive or Self-embedded Sources

It is now quite natural to allow a grammar to have itself as a sub-grammar or to allow a grammar M_1 to contain a grammar M_2 which, in its turn, contains M_1, and so on. The grammars thus obtained cannot, however, be rewritten as an ordinary Markov grammar. The relation between an ordinary Markov grammar and a recursive one is exactly similar to the relation between Finite State Languages and Phrase Structure Languages.

To be more precise, assume that we have a set of Markov grammars M'_0, M'_1, ..., M'_N where M'_0 is called the main grammar and in the sense that the process always starts at the initial state in M'_0 and ceases when it reaches the final state in M_0. Each of the grammars may contain any number of the others (and itself) as sub-grammars. The only restriction is that from any state in any one of the grammars there should exist a path which ends up at the final state of M_0.

Remark

If we interpret a source of our kind as a Phrase Structure Language, the rewriting rules are all of the following kind:

$$(5) \qquad S_i \rightarrow A_{ik} + S_k \qquad \underline{\text{or}} \qquad S_n \rightarrow \#;$$

where the S's are all non-terminal symbols. (They stand for the names of the states in the sources - M'_0, M'_1, ..., M'_N and where S_0 is assumed to be the initial symbol /the Chomskyan S/ and S_n is the terminating state which produces the sentence delimiter #. The symbols A_{ik} are either terminal symbols /letters from a finite alphabet/ or non-terminal symbols equal to the name of the initial state in one of the grammars M'_0, M'_1, ..., M'_N /one may also say that A_{ik} stands as an abbreviation for an arbitrary sentence of that grammar/.)

We associate each grammar M'_j with the grammar M_j, $j = 0, 1, \dots, N$, by just considering it as a non-recursive one, that is, we consider all the symbols A_{ik} as terminal symbols (even if they are not). The grammars thus obtained are ordinarily Markov grammars according to our definition, and the entropies $H_j = H(M_j)$ are easily computed according to formula (1), as are the stationary distributions /formula (2)/. The follwoing theorem shows how the entropies H'_j for the fully recursive grammars M'_j are connected with the numbers H_j.

Theorem

The entropy H'_j for a set of recursive Markov grammar M'_j, $j = 0, 1, \dots, N$, can be calculated according to the formula

$$(6) \qquad H'_j \{1 + \sum_k y_{jk} (\mu_k - 1)\} - \sum_k y_{jk} \mu_k H'_k = H_j$$
$$j = 0, 1, \dots, N.$$

Here the factors y_{jk} are dependent only of the probability matrix of the grammar and the numbers μ_k defined as the mean sentence length of the sentences of the grammar M'_k, $k = 0, 1, \dots, N$, and computable according to lemma below.

H'_0 is the entropy for the grammar.

The theorem above is a direct application for the grammar of formula (4), sec. 2.

The coefficients y_{jk} in formula (6) can, more precisely, be calculated as a sum of terms of the type $x_i p_{im}$ with the indices (i, m) are where the grammar M'_k appears in the grammar M'_j; x_i and p_{im} are the components the stationary distribution and probability matrix for the grammar M_j.

Assume now that we have a Markov grammar of our type, but for which each transition will take a certain amount of time. A very natural question is then: "What is the expected time to produce a sentence in that language?" The answer is in the following lemma.

Lemma

Let M be a Markov grammar with states S_i, $i = 0, 1, \ldots, n$, where S_0 and S_n are the initial and final states respectively.

Assume that each transition $S_i \rightarrow S_k$ will take y_{ik} time units.

Denote the expected time for arrival at S_n given that the grammar is in state S_i by t_i, $i = 0, 1, \ldots, n$, (thus t_0 is the expected time for producing a sentence). The times t_1 will then fulfill the following set of simultaneously linear equations:

$$(7) \qquad t_i = \sum_k p_{ik} (t_{ik} + t_k)$$

Formula (7) is itself a proof of the lemma.

With more convenient notations we can write (7) as

$$(E - P)\, t = p_t$$

where E is the unit matrix, P is the probability matrix (with $P_{nn} = 0$) and p_t is the vector with components

$$p_i\,(t) = \sum_m p_{im}\, t_{im}, \quad i = 0, 1, \ldots, n.$$

The application of the lemma for computing the numbers μ_k in formula (6) is now the following.

The transition times of the lemma are, of course, the expected time (or "lengths" as we have called it earlier) for passing via a sub-grammar of the grammar under consideration. Thus the number t_{ik} is itself the unknown entities μ_k.

For each of the sub-grammars $M_j^{!}$, j = 0, 1, ..., N, we get a set of linear equations of type (7) for determining the vectors t of lemma. The first component of this vector, i.e., the number t_0, is then equal to the expected length, μ, of the sentences of that grammar. (Unfortunately, we have to compute extra the expected time for going from any state of the sub-grammars to the corresponding final state.)

The total number of unknowns involved when computing the entropy of our grammar (i.e., the entropy H_0') is equal to

(the total number of states in all our sub-grammars) plus
(the number of sub-grammars).

This is also the number of equations for we have n + 1 equations from formula (6) and then (n + 1) sets of equations of the type (7). We assert that all these simultaneous equations are solvable, if the grammar fulfills the conditions we earlier stated for the grammar, i.e., that from each state in any subgrammar exists at least one path to the final state of that grammar.

Catégories Morphologiques et analyse contextuelle dans la linguistique algébrique

par Solomon Marcus

Une caractéristique du développement récent de la linguistique algébrique c'est l'existence de maintes théories parallèles, ayant pour objet une même réalité linguistique ou des faits assez semblables. Mais pour la plupart de ces théories on ne connaît pas les rapports exacts et on continue d'utiliser même des terminologies différentes pour des notions identiques, ce qui n'est pas de nature à favoriser le progrés de cette science. C'est justement la situation dans les études d'analyse algébrique contextuelle. Nous nous proposons de discuter, dans ce qui suit, les notions introduites par Dobrušin [1] , [2] Sestier [11] et Sakai [10], notions qui s'avèrent intimement liées, parfois même équivalentes.

Considérons un vocabulaire fini V . Toute suite finie d'éléments de V est une phrase (sur V) ; Toute collection de phrases (sur V) est un langage (sur V). Il sera sous-entendu le fait que toutes les notions sont relatives à V et un langage L sur V et nous allons supprimer la référence explicite à ce fait.

Soient $a \in V$ et $b \in V$. Soit L un langage sur V . On dit que a domine b et on écrit $a \longrightarrow b$ si pour tout couple de phrases f et g la relation $fa\;g \in L$ implique la relation $fbg \in L$. Si L est une langue naturelle, la relation $a \longrightarrow b$ a l'interprétation suivante. L'homonymie morphologique qui se manifeste parmi les formes flexionnelles du mot a est plus pauvre ou égale à celle qui se manifeste parmi les formes flexionnelles du mot b . Par exemple, si V est le vocabulaire du français écrit et si L est la collection des phrases correctes du français écrit, alors, en prenant a = petit et b = mince , on constate que $a \longrightarrow b$, mais on n'a pas $b \longrightarrow a$ et ce fait est dû à ce que à des formes flexionnelles différentes de mince correspondent des formes flexionnelles différentes de petit, mais la réciproque n'est pas vraie, car aux formes petit et petite correspond la même forme mince. Si l'on a, à la fois, $a \longrightarrow b$ et $b \longrightarrow a$, on écrit $a \longleftrightarrow b$ et il est facile de voir que $\longleftrightarrow$ est une relation d'équivalence dans V . La classe d'équivalence de a sera notée par $S(a)$; c'est la classe de distribution du mot a .

Par exemple, dans l'exemple du français écrit envisagé ci-dessus, on a *petit* $\longleftrightarrow$ *grand*, donc *grand* $\in$ S(*petit*). Un mot *a* est dit initial si pour $\underline{b} \longrightarrow \underline{a}$ on a toujours $\underline{b} \in S(a)$. R.L. Dobrušin a introduit la notion de catégorie morphologique (Dobrušin dit : grammaticale) élémentaire engendrée par un mot initial *a* ; c'est l'ensemble $\mathcal{G}(a) = \{b \; ; \; a \longrightarrow b\}$. L'interprétation de cette notion est la suivante : $\mathcal{G}(a)$ est l'ensemble des mots qui possèdent toutes les valeurs morphologiques de *a*. Par exemple, dans le français écrit *a* = *petit* est un mot initial, tandis que $\mathcal{G}(a) = \{$ *petit*, *grand*; *mince*, *gros*, ... $\}$ est l'ensemble des adjectifs qualificatifs qui sont au masculin singulier. Naturellement, il peut arriver que certains mots de $\mathcal{G}(a)$ possèdent, en outre, des valeurs étrangères à *a*, comme on constate aussi sur l'exemple ci-dessus : le mot *mince* possède aussi la valeur du féminin, tandis que le mot *gros* possède aussi la valeur du pluriel.

Mais la notion de catégorie morphologique élémentaire ne couvre pas toutes les catégories morphologiques de la linguistique. Par exemple, la catégorie du singulier, celle du féminin, celle de l'adjectif qualificatif et beaucoup d'autres ne correspondent à aucune catégorie morphologique élémentaire. Ce fait nous a déterminé d'introduire une certaine extension de la notion de Dobrušin (voir [3], [4] ; [5] et le chapitre V de [8]). Si $A \subseteq V \supseteq B$ et si $\underline{a} \longrightarrow \underline{b}$ pour chaque $\underline{a} \in \underline{A}$ et pour chaque $\underline{b} \in \underline{B}$, on dit que *A* domine *B* et on écrit $\underline{A} \longrightarrow \underline{B}$. Posons $A_1 = \{a; A \longrightarrow \underline{a}\}$. L'ensemble $\mathcal{G}(A) = A \cup A_1$ est, par définition, la catégorie morphologique engendrée par A. Dans le cas particulier où $A = \{\underline{a}\}$ et *a* est un mot initial, on obtient la notion de catégorie morphologique élémentaire, due à Dobrušin. L'intérêt linguistique de la notion que nous venons de définir vient du fait qu'elle couvre toutes les catégories morphologiques intuitives de la linguistique. Ce fait est la conséquence des deux propositions suivantes :

1° - La réunion de deux catégories morphologiques est aussi une catégorie morphologique (voir la Proposition de [4], P. 328, où les catégories morphologiques sont appelées catégories grammaticales).

2° - Toute catégorie morphologique intuitive correspond à une réunion de catégories morphologiques élémentaires (ce qui justifie la dénomination de catégorie morphologique élémentaire ; c'est l'atome morphologique, la cellule qui se trouve à la base de toute catégorie morphologique). Prenons, par exemple, la catégorie morphologique du singulier des adjectifs qualificatifs du français écrit. Les mots appartenant à cette catégorie forment un ensemble qui n'est autre chose que la réunion des catégories morphologiques élémentaires $\mathcal{G}$(petit) et $\mathcal{G}$ (petite). En ce qui concerne la catégorie morphologique des adjectifs qualificatifs, elle est une réunion de quatre catégories morphologiques élémentaires, à savoir $\mathcal{G}$(petit) , $\mathcal{G}$ (petite) , $\mathcal{G}$ (petits) , $\mathcal{G}$ (petites) . En ce qui concerne la réunion de deux catégories morphologiques élémentaires distinctes, elle n'est jamais une catégories morphologique élémentaire. En effet, soient $\mathcal{G}$ (a) et $\mathcal{G}$(b) deux catégories morphologiques élémentaires distinctes, et supposons qu'il existe un mot initial c tel que $\mathcal{G}(c) = \mathcal{G}(a) \cup \mathcal{G}(b)$ (les mots a et b sont aussi initiaux, par la définition même de la notion de catégorie morphologique élémentaire). On a donc $c \longrightarrow a$ et $c \longrightarrow b$. Cela implique, en vertu du fait que les mots a et b sont initiaux, $a \in S(c)$ et $b \in S(c)$, donc $a \longleftrightarrow b$. Mais on obtient ainsi une contradiction avec l'hypothèse que $\mathcal{G}(a) \neq \mathcal{G}(b)$; donc la réunion de $\mathcal{G}$ (a) et $\mathcal{G}$ (b) n'est plus une catégorie morphologique élémentaire. Mais cette réunion est toujours (en vertu du 1°) une catégorie morphologique, ce qui prouve l'avantage de cette dernière notion.

Passons maintenant au point de vue de Sestier [11] qui sera présenté ici avec certaines modifications de terminologie et de notation. Tout couple ordonné $\langle f,g \rangle$ de phrases (sur V) est un contexte (sur V). On dit que le contexte $\langle f,g \rangle$ accepte la phrase h (par rapport à un langage L sur V) si $fgh \in L$. Désignons par $\alpha(h)$ l'ensemble des contextes qui acceptent la phrase h et par $\alpha^{-1}(\gamma)$ l'ensemble des mots acceptés par le contexte γ . Soit maintenant un ensemble X V , et un ensemble Y de contextes. Posons

$$\alpha(X) = \bigcap_{x \in X} \alpha(x) \quad , \quad \alpha^{-1}(Y) = \bigcap_{y \in Y} \alpha^{-1}(y)$$

$$\varphi(X) = \alpha^{-1}(\alpha(X)) \quad \text{et} \quad \Psi(Y) = \alpha(\alpha^{-1}(Y))$$

L'ensemble φ (X) est la fermeture de Sestier de l'ensemble X , tandis que Ψ (Y) est la fermeture de Sestier de l'ensemble Y .

Dans le premier cas, la fermeture est un ensemble de mots, tandis que dans le deuxième elle est un ensemble de contextes.

Comme nous l'avons montré ailleurs (propositions 1, 2 et 3 de [7] , on a, pour tout ensemble X de mots, l'inclusion $\mathcal{G}(X) \subseteq \varphi(X)$; il y a des cas où l'inclusion est stricte, mais si X se réduit à un seul mot, on en a toujours l'égalité. En particulier, il s'ensuit que la catégorie morphologique élémentaire engendrée par un mot initial coïncide avec la fermeture de Sestier de ce mot.

Il est intéressant de faire une confrontation entre les catégories morphologiques et les fermetures de Sestier car les unes et les autres ont des significations linguistiques très importantes. Voici d'abord quelques analogies (X est une partie quelconque de V) :

a) $X \subseteq \mathcal{G}(X)$;	a') $X \subseteq \varphi(X)$;
b) $\mathcal{G}(\mathcal{G}(X)) = \mathcal{G}(X)$;	b') $\varphi(\varphi(X)) = \varphi(X)$;
c) $V - \mathcal{G}(X) \subseteq \mathcal{G}(V - X)$;	c') $V - \varphi(X) \subseteq \varphi(V - X)$;
d) il y a des cas où l'inclusion a) est stricte et il y en a d'autres où l'on a l'égalité ;	d') il y a des cas où l'inclusion a') est stricte et il y en a d'autres où l'on a l'égalité ;
e) il y a des cas où l'inclusion c) est stricte et il y en a d'autres où l'on a l'égalité ;	e') il y a des cas où l'inclusion c') est stricte et il y en a d'autres où l'on a l'égalité ;
f) afin que X soit une catégorie morphologique il faut et il suffit que $\mathcal{G}(X) = X$	f') afin que X soit une fermeture de Sestier il faut et il suffit que $\varphi(X) = X$

Les propositions a) et a') sont évidentes. La proposition b) est le théorème 5 de [4] , tandis que la proposition b') a été donnée dans [12] (voir aussi la proposition 18 de [12]). Les propositions c) et c') sont, respectivement, les propositions 23 et 19 de [12] . Les propositions d) et d') sont évidentes. Les propositions e) et e') résultent respectivement des propositions 24 et 20 de [12] . Les propositions f) et f') sont respectivement des conséquences de b) et b').

Il y a probablement maintes autres analogies entre $\mathcal{G}$ et φ et on pourrait chercher ainsi l'analogue pour φ des théorèmes de [3], [4], [5] et du chapitre V de [8]. Mais nous voulons attirer l'attention sur quelques propriétés qui montrent les différences de nature algébrique et de signification linguistique des deux opérateurs $\mathcal{G}$ et φ (A et B sont deux sous-ensembles de V) :

g) il y a des cas où $A \subset B$; mais aucun des ensembles $\mathcal{G}(A)$ et $\mathcal{G}(B)$ n'est contenu dans l'autre ;	g') $A \subseteq B \Longrightarrow \varphi(A) \subseteq \varphi(B)$;
h) la réunion de deux catégories morphologiques est aussi une catégorie morphologique ;	h') la réunion de deux fermetures de Sestier n'est pas toujours une fermeture de Sestier
i) l'intersection de deux catégories morphologiques n'est pas toujours une catégorie morphologique ;	i') l'intersection de deux fermetures de Sestier est aussi une fermeture de Sestier ;
j) il y a des cas où $\mathcal{G}(A \cap B)$ n'est pas contenu dans $\mathcal{G}(A) \cap \mathcal{G}(B)$;	j') $\varphi(A \cap B) \subseteq \varphi(A) \cap \varphi(B)$;
k) il y a des ensembles A et B tels que $\mathcal{G}(A \cup B) \subset \mathcal{G}(A) \cup \mathcal{G}(B)$	k') $\varphi(A) \cup \varphi(B) \subseteq \varphi(A \cup B)$
l) Si $A - B \neq 0 \neq B - A$, alors $\mathcal{G}(A) = \mathcal{G}(B)$ si et seulement s'il existe un mot a tel que $A \cup B \rightarrow S(a) \supseteq A \Delta B$, où Δ signifie la différence symétrique.	l') $\varphi(A) = \varphi(B)$ si et seulement si $\alpha(A) = \alpha(B)$.

La proposition g) résulte de l'exemple suivant. Soit $V = \{a, b, c, d, e\}$, $L = \{abc, adc, ebc, edc\}$, $A = \{a\}$, $B = \{a, b, c\}$. On a $A \subset B$, $\mathcal{G}(A) = \{a, e\}$, $\mathcal{G}(B) = \{a, b, c\}$, donc aucun des ensembles $\mathcal{G}(A)$ et $\mathcal{G}(B)$ n'est contenu dans l'autre. La proposition g') est la proposition 12 de [12]. La proposition h) a été établie dans [4], p. 327-328. Quant à la proposition h'), il faut remarquer qu'elle est équivalente (en vertu de la

proposition b')) à l'existence de deux ensembles A et B qui ne satisfont pas l'égalité $\varphi(\varphi(A) \cup \varphi(B)) = \varphi(A) \cup \varphi(B)$. C'est justement le cas, comme le montre l'exemple suivant, dû à Grzegorz Rozenberg (communication orale).
Soit $V = \{a, b, c, d, e, f, g\}$, $L = \{cac, cdc, gbg, geg, cfc, gfg\}$.
Posons $A = \{a, d\}$, $B = \{b, e\}$. On a $\varphi(A) = \{a, d, f\}$, $\varphi(B) = \{b, e, f\}$
$\varphi(A) \cup \varphi(B) = \{a, b, d, e, f\}$,
$\varphi(\varphi(A) \cup \varphi(B)) = \alpha^{-1}(\alpha(\{a, b, d, e, f\})) = \alpha^{-1}(0) = V$

Ce qui rend plus intéressant cet exemple c'est le fait que A et B sont ici des classes de distribution ; en effet, on a $A = S(a)$ et $B = S(b)$. Il s'ensuit donc que, non seulement la réunion de deux fermetures de Sestier n'est pas toujours une fermeture de Sestier, mais elle ne l'est ni même dans le cas particulier où les deux fermetures envisagées sont des catégories morphologiques élémentaires (on sait, d'après la proposition 3 de [7] , que $\mathcal{G}(S(x)) = \varphi(S(x))$, quel que soit le mot x). Mais nous avons vu que les réunions finies de catégories morphologiques élémentaires donnent justement les catégories morphologiques intuitives des langues naturelles ; il s'ensuit donc que ces catégories intuitives ne correspondent pas toujours à des fermetures de Sestier, ce qui met la notion de fermeture due à Sestier dans une situation d'infériorité par rapport à la notion de catégorie morphologique (si l'on envisage le point de vue morphologique). D'autre part, du point de vue de l'analyse contextuelle, c'est la fermeture de Sestier qui semble plus naturelle et plus utile.

La proposition i) résulte du même exemple utilisé pour prouver la proposition g) . En effet, on a, dans cet exemple, $\mathcal{G}(A) \cap \mathcal{G}(B) = \{a\}$, mais il existe aucun ensemble C tel que $\mathcal{G}(C) = \{a\}$. En effet, on a $\mathcal{G}(\{a\}) = \{a, e\}$, donc $\mathcal{G}(\{a\}) \neq \{a\}$ et, en vertu de la proposition f), $\{a\}$ n'est pas une catégorie morphologique. En ce qui concerne la proposition i') , il faut remarquer qu'elle est équivalente à l'assertion $\varphi(\varphi(A) \cap \varphi(B)) = \varphi(A) \cap \varphi(B)$. Voici une démonstration rapide de cette égalité, due à R. S. Waligorski (communication orale). En utilisant les propositions j') et b') , on a $\varphi(\varphi(A) \cap \varphi(B)) \subseteq \varphi(\varphi(A)) \cap \varphi(\varphi(B)) = \varphi(A) \cap \varphi(B)$. D'autre part, en vertu de la proposition a'), on a $\varphi(A) \cap \varphi(B) \subseteq \varphi(\varphi(A) \cap \varphi(B))$.
La proposition j) a été établie par E. Toma Marcu (voir la proposition 22 du [12]).
Le même auteur a démontré la proposition j') (voir la proposition 13 de [12]).
La proposition k) résulte de l'exemple suivant. Soient $V = \{a, b, c, d, e, f, g, h,\}$

$L=\{abc, adc, efg, ehg\}$. Posons $A=\{b\}$, $B=\{f\}$. On a $\mathcal{G}(A)=\{b, d\}$, $\mathcal{G}(B)=\{f, h\}$, $\mathcal{G}(A\cup B)=\mathcal{G}(\{b, f\})=\{b, f\}$, donc $\mathcal{G}(A\cup B)\subset\mathcal{G}(A)\dot{\cup}\mathcal{G}(B)$
La proposition k') est justement la proposition 15 de [12]. La proposition l) a été démontrée dans [3] (voir aussi le chapitre V de [8]) tandis que la proposition l') est la proposition 18 de [12] .

On constate donc qu'il y a une certaine dualité entre les opérateurs G et φ , en ce qui concerne les opérateurs de réunion et d'insertion (voir les propositions h) et h'), i) et i')) mais en général l'opérateur φ présente une certaine supériorité de nature algébrique (voir, par exemple, la propriété de monotonie de l'opérateur φ). Il serait intéressant d'étudier le comportement réciproque de ces deux opérateurs, par exemple leur superposition, la distributivité de l'un par rapport à l'autre, etc...

Dans un travail récent, Itiroo Sakai développe une analyse contextuelle qui en beaucoup de points s'approche des notions ci-dessus, dès que ces notions sont envisagées non seulement pour les mots (c'est-à-dire pour les éléments de V), mais aussi pour les phrases (c'est-à-dire pour les éléments du langage universel sur V), il introduit la notion de voisinage complet d'une phrase f, qui revient à l'ensemble $\mathcal{A}(f)$, et la notion de voisinage élémentaire, qui revient à un voisinage complet qui est une classe de distribution, c'est-à-dire qui est de la forme S(f). Dans [8] nous avons établi certaines propositions qui lient les notions de Sakai et celles ci-dessus, mais il reste encore à approfondir cette liaison. Des aspects nouveaux de certaines des notions ci-dessus sont étudiés par Revzin [9] et par Marcus [6] .

Bibliographie

1. R.L. Dobrušin Elementarnaja grammatičeskaja kategorija
Bjulleten Obdeinenija po probleman mašinnago perevoda, 1957, nr. 5, p. 19-21

2. R.L. Dobrušin Matematičeskie metody v lingvistike. Priloženie Matematičeskoe prosvescenie, vol. 6, 1961 p. 52-59

3. S. Marcus sur un modèle logique de la catégorie grammaticale élémentaire, I. Revue de mathématiques pures et appliquées, vol. 7, 1962; nr. 1 , p 91-107

4. S. Marcus Sur un modèle logique de la catégorie grammaticale élémentaire, II. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, vol. 8, 1962, nr. 3-4, p. 323-329.

5. S. Marcus Ob odnoi logičeskoi modeli elementarnoi grammatičeskoi kategorii, III, Revue de Mathématiques pures et appliquées, vol. 7, 1962, nr. 4., p. 683-691.

6. S. Marcus Analyse contextuelle. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung, vol. 18, 1965, nr. 3, p. 3ol-313

7. S. Marcus Catégories de Dobrušin, fermetures de Sestier et voisinages de Sakai, Glosse, vol. 1, 1967, nr.1.

8. S. Marcus Introduction mathématique à la linguistique structurale. Dunod, Paris; 1967

9. I.I. REVZIN Metod modelirovanija i typologia slavianskich jazykov, Izd. Nauka, Moskva, 1967

10. I. Sakai Some mathematical aspects of syntactic description (Preprint). International Conference of Computational linguistics, New-York, 1965, nr.21

11. A.Sestier Contribution à une théorie ensembliste des classifications linguistiques, actes du Premier Congrés de l'AFCAL, 1960 Paris, 1961, P. 293-305

12. E. Toma Marcu Propiretăţi ale închiderilor Sestier din teoria algebrică a gramaticii. Studii şi cercetări matematice vol.12, 1967.